Федеральное государственное бюджетное учреждение высшего образования
«Санкт-Петербургский государственный университет»

Факультет прикладной математики – процессов управления

Итоговый проект по дисциплине «Неклассические логики»:

«Проектирование и реализация системы нечеткой логики: Оценка необходимости увеличения аппаратных ресурсов веб-сервера при однократном увеличении показателя Load Average»

Выполнил:
Докиенко Денис Александрович
Группа:
17.Б13-пу

Санкт-Петербург

2020

Оглавление

Введение

Логики, отличные от бинарной (т.н. «Булевой логики»), в настоящее время принято определять единым термином «Неклассические логики». Одной из таких логик является «Нечеткая логика», базирующаяся на теории нечетких множеств. Она позволяет определять принадлежность элемента к множеству не классическим бинарным значением (1 - принадлежит, 0 - не принадлежит), а любым вещественным значением из интервала [0,1]. Таким образом, опуская весь математический аппарат, обосновывающий теорию нечетких логик, можно сказать, что нечеткая логика позволяет определять степень истинности того или иного высказывания. К примеру, в высказывании “вода горячая” можно ввести степень теплоты воды, изменяя степень истинности высказывания. Если высказывание истинно на 0,75, то вода скорее горячая, чем холодная (теплая).

Таким образом, нечеткая логика близка к логике человеческого мышления куда больше, чем Булева логика, используемая при проектировании ЭВМ.

На этой идее основана теория нечеткого управления, позволяющая имитировать экспертную оценку ситуации, используя последовательность человеческих суждений, сформулированных в форме, близкой к форме естественных языков. Нечеткое управление хорошо подходит для работы с системами, в которых фактор неопределенности или иного рода неточности играет большую роль.

Важно отметить, что, согласно Fuzzy Approximation Theorem, любая математическая система может быть аппроксимирована системой, основанной на нечеткой логике.

Целью данной работы является построение модели автоматизированной системы нечеткого управления для оценки необходимости увеличения аппаратных ресурсов веб-сервера при однократном не постоянном увеличении показателя средней загрузки Unix-подобных операционных систем (“Load Average”). Для разработки используется среда Matlab Fuzzy Logic Toolbox.

Постановка задачи

Любое развивающееся веб-приложение сталкивается с проблемой корректного обслуживания возросшего числа посетителей (“highload”). Проблемы могут выражаться в:

- возникновении ошибок 5xx (тип ошибок, относящийся к ошибкам на сервере)
- полном отказе системы
- уменьшении скорости работы приложения

Причиной при этом могут быть:

- неправильно настроенные параметры ПО сервера (параметры веб-сервера, сервера БД и т.п.)
- ошибки в работе самого веб-приложения или плохо выполненная оптимизация (плохо написанные запросы к БД приложения и т.п.)
- недостаток аппаратных мощностей системы (CPU, RAM, скорость работы ПЗУ, ширина сетевого канала до сервера и т.п.)

Целью данной работы является разработка системы, определяющей необходимость увеличения ресурсов RAM, CPU и скорости записи/чтения диска. При этом дать гарантию, что увеличение аппаратных мощностей решит проблему highload невозможно. Система исключительно должна уметь диагностировать необходимость увеличения аппаратных ресурсов, не рассматривая возможность некорректной настройки ПО сервера или ошибок в коде приложения.

В основе работы разрабатываемой системы лежит оценка загрузки сервера посредством параметра Load Average. Данный параметр показывает общую загрузку системы. Упрощая можно сказать, что он отражает количество процессов, находящихся в состоянии ожидания ресурсов, выделяемых операционной системой. В системе Load Average как правило представлен в виде трех усредненных значений (экспоненциальное скользящее среднее) Load Average за последние: 1 минуту, 5 минут и 15 минут.

Систему можно использовать при возникновении ситуации разового резкого увеличения Load Average сервера, на котором размещено веб-приложение. С ожидаемым сценарием использования системы можно ознакомиться в разделе “Анализ системы”.

Математическое обоснование

В системах, построенных на нечеткой логике, принадлежность элемента множеству описывается функцией принадлежности. Она указывает, в какой степени элемент принадлежит нечеткому множеству и может принимать любые значения в интервале [0,1].

Входные и выходные переменные в системах основанных на нечеткой логике называются лингвистическими переменными. Такие переменные могут принимать значение фраз на естественном языке. Набор этих фраз называется терм-множеством лингвистической переменной.

В качестве схемы нечеткого вывода в данной работе используется алгоритм Мамдани. Он состоит из следующих последовательно выполняющихся этапов:

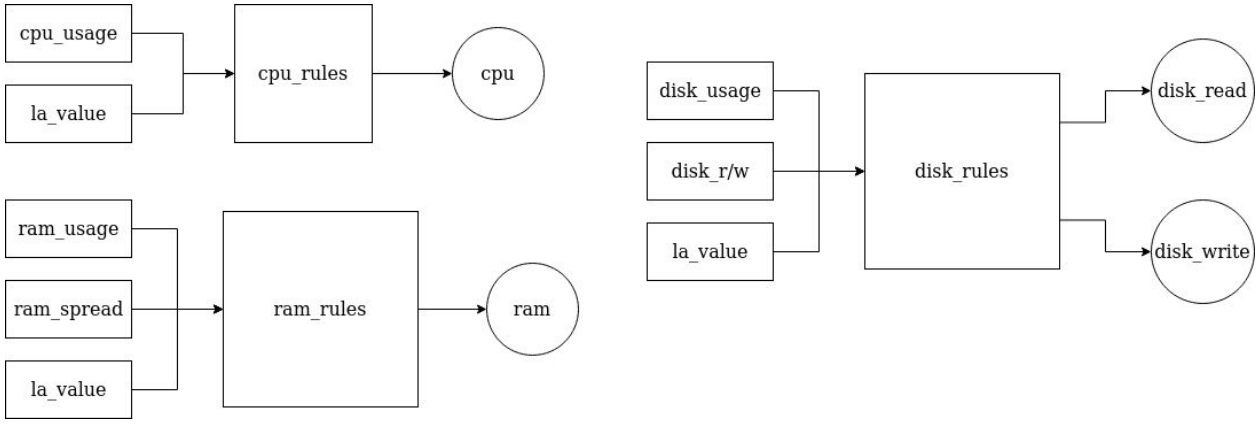
1. Формирование базы правил
2. Фаззификация входных переменных
3. Агрегирование подусловий
4. Активизация подзаключений
5. Аккумуляция заключений
6. Дефаззификация выходных переменных

Со стороны разработчика подобной системы проектирование архитектуры можно разделить на три этапа:

- Выбор входных и выходных переменных системы и определение их взаимосвязей
- Разработка набора правил
- Формирование функций принадлежности для входных и выходных переменных (правил фаззификации и дефаззификации)

Описание переменных и архитектуры системы

В системе используются 6 входных и 4 выходных переменных. Поскольку аппаратные ресурсы (RAM, CPU, диск) независимы друг от друга, на каждую выходную переменную влияет только некоторая часть входных переменных. Таким образом, система может быть представлена как набор из трех различных систем:

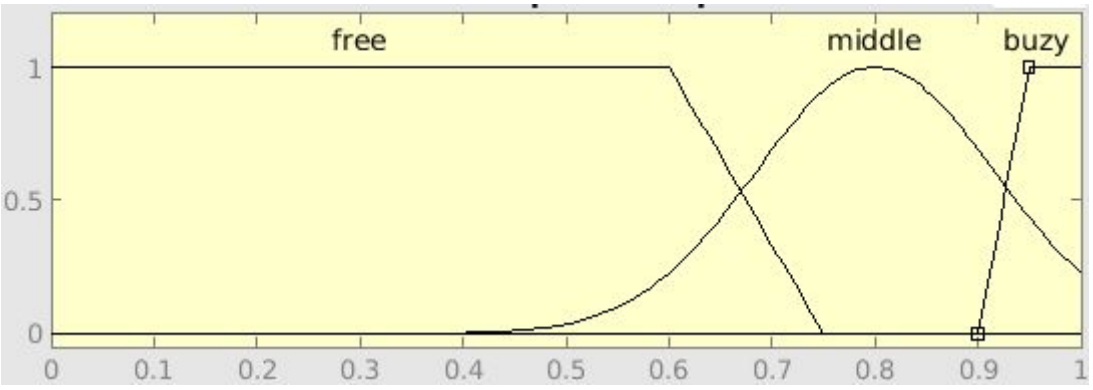


Опишем используемые в системе входные переменные и представим их функции принадлежности:

1. cpu_usage

Отражает процент использованных ресурсов CPU в момент пиковой загрузки системы. Определяется как частное использованных ресурсов CPU к максимально доступным ресурсам CPU. Терм-множество:

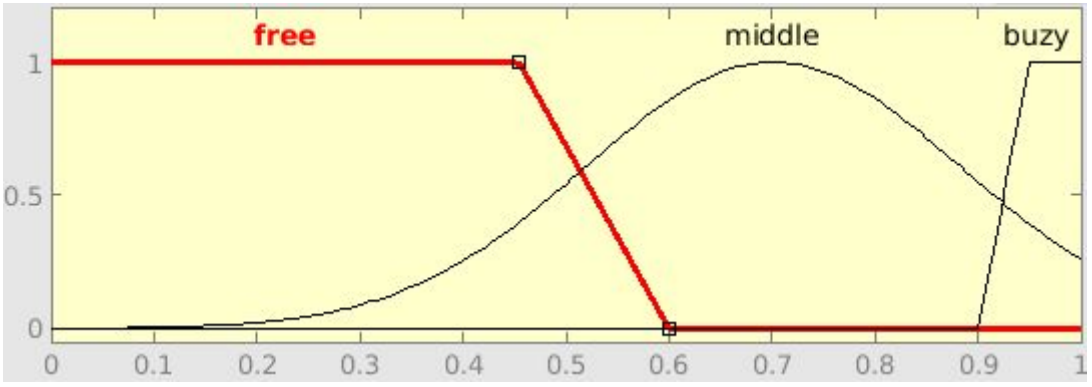
- free - CPU нагружен слабо
- middle - средний уровень загрузки CPU
- buzy - CPU сильно нагружен



2. ram_usage

Отражает процент использованных ресурсов RAM в момент пиковой загрузки системы. Определяется как частное использованных ресурсов RAM к максимально доступным ресурсам RAM. Терм-множество:

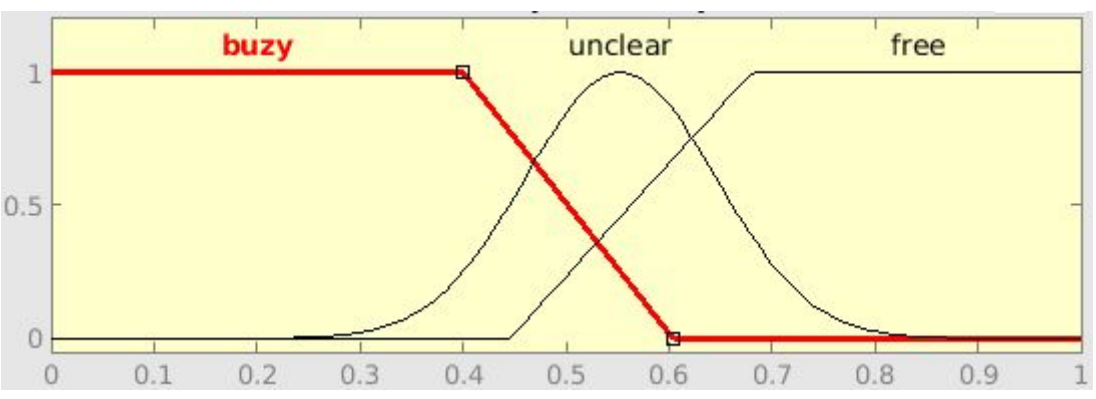
- free - RAM используется не сильно
- middle - RAM используется на среднем уровне
- buzy - RAM используется почти полностью или полностью



3. ram_spread

Показывает, какое количество свободной RAM обычно доступно в системе. Параметр введен по причине того, что некоторое серверное окружение “блокирует” всю возможную RAM. В системе такая память выглядит как память, занятая одним из процессов, но фактически она не используется, будучи зарезервированной “на будущее”. Так ведет себя, например, Bitrix Virtual Machine. Вычисляется переменная как частное среднего значения свободной RAM в периоды с низким значением Load Average к максимально доступному значению RAM. Терм-множество:

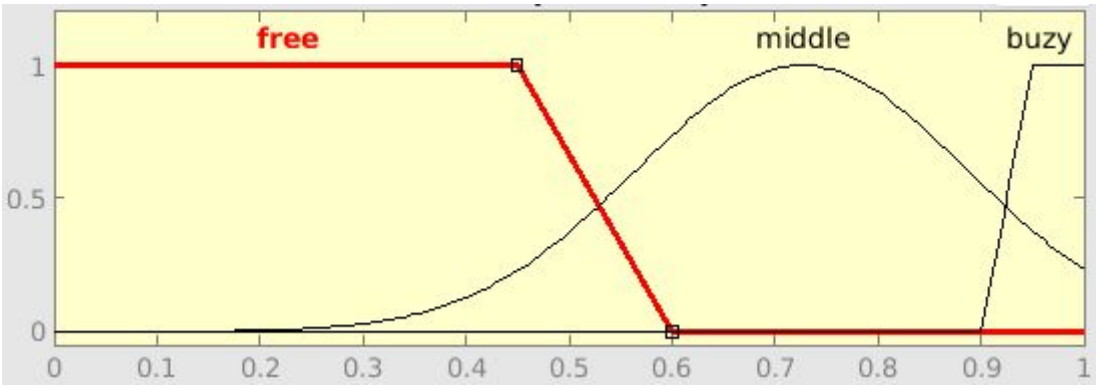
- free - обычно в системе доступна большая часть RAM, т.е. вероятно, что окружение резервирующее RAM не используется
- unclear - обычно в системе доступна примерно половина RAM, поэтому нельзя точно диагностировать наличие или отсутствие описанного ПО
- buzy - обычно в системе используется большая часть RAM, поэтому вероятно наличие окружения резервирующего RAM



4. disk_usage

Отражает процент использованных ресурсов диска в момент пиковой загрузки системы. Определяется как частное использованных ресурсов диска к максимально доступным ресурсам диска. Терм-множество:

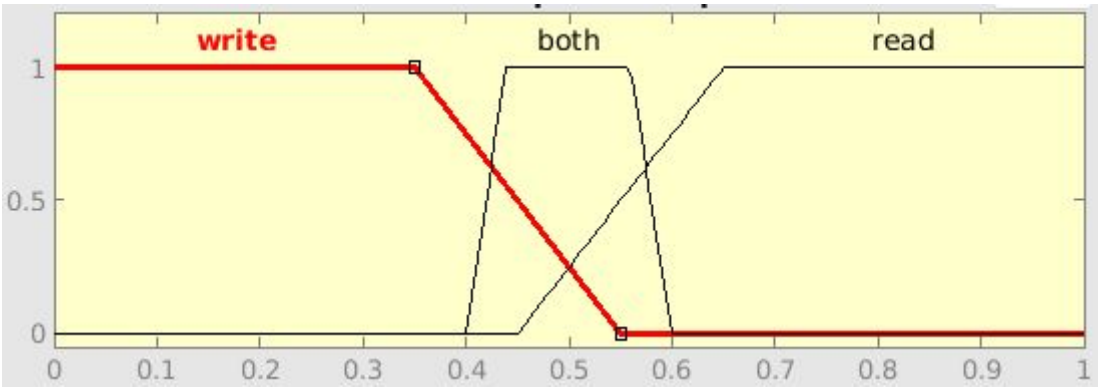
- free - диск используется не сильно
- middle - средний уровень загрузки диска
- buzy - диск используется почти полностью или полностью



5. disk_r/w

Показывает, по какому именно показателю загружен диск: по чтению или по записи. Определяется как доля ресурсов диска используемых для чтения. Терм-множество:

- write - большая часть ресурсов диска используется для записи на диск
- both - ресурсы диска используются примерно одинаково для чтения и для записи
- read - большая часть ресурсов диска используется для чтения с диска

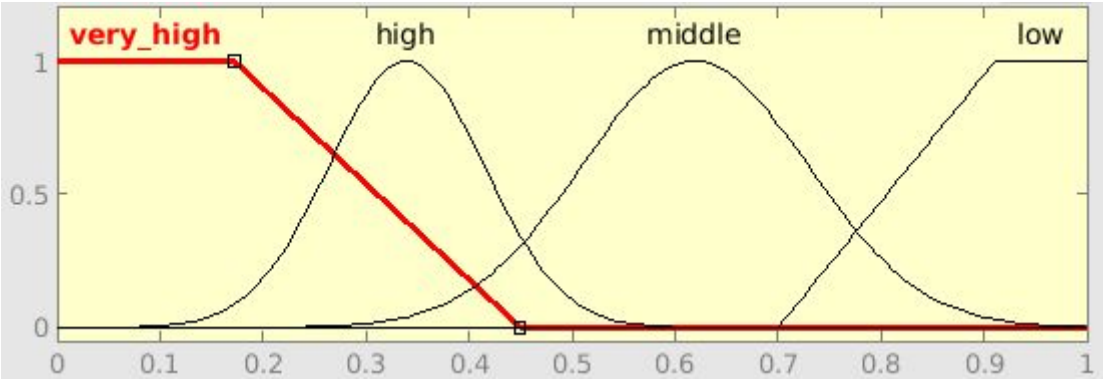


6. la_value

Показывает, насколько сильно увеличивается Load Average в момент пиковой загрузки системы, относительно обычного показателя Load Average на сервере.

Определяется как отношение Load Average до момента начала резкого возрастания Load Average к пиковому значению Load Average. Терм-множество:

- very_high - значение Load Average очень сильно возрастает в момент пиковой нагрузки
- high - значение Load Average довольно сильно возрастает в момент пиковой нагрузки
- middle - значение Load Average не сильно возрастает в момент пиковой нагрузки
- low - значение Load Average почти не возрастает в момент пиковой нагрузки

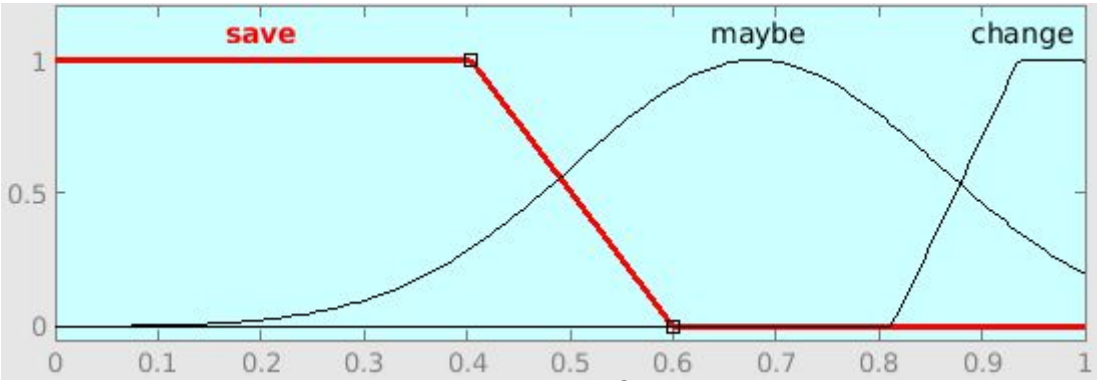


Опишем используемые в системе выходные переменные и представим их функции принадлежности:

1. cpu

   Показывает, стоит ли увеличивать ресурсы CPU. Терм-множество:

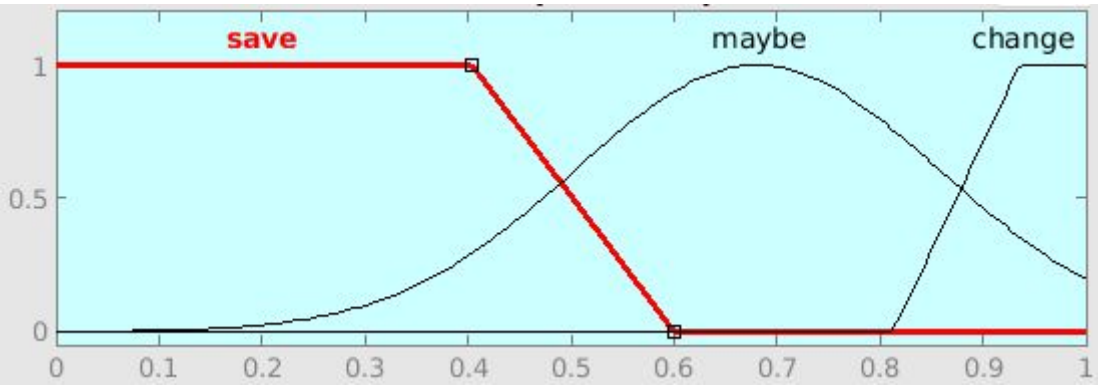
   - save - увеличивать ресурсы не нужно
   - maybe - пока увеличивать ресурсы не требуется, но в обозримом будущем это необходимо будет сделать. Возможно, стоит увеличить ресурсы сейчас, чтобы предотвратить их недостаток в будущем
   - change - необходимо увеличить ресурсы CPU. Возможно, стоит подумать про улучшение CPU (покупка CPU с меньшим тех. процессом, большей тактовой частотой, большим количеством ядер) или про включение Intel Turbo Boost или аналогичных технологий



2. ram

Показывает, стоит ли увеличивать ресурсы RAM. Терм-множество:

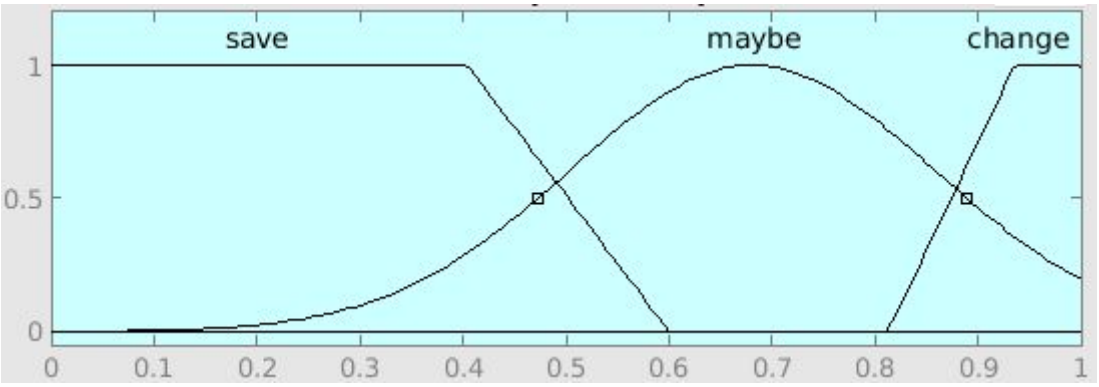
- save - увеличивать ресурсы не нужно
- maybe - пока увеличивать ресурсы не требуется, но в обозримом будущем это необходимо будет сделать. Возможно, стоит увеличить ресурсы сейчас, чтобы предотвратить их недостаток в будущем
- change - необходимо увеличить размер RAM



3. disk_read

Показывает, стоит ли увеличивать ресурсы диска на чтение. Терм-множество:

- save - увеличивать ресурсы не нужно
- maybe - пока увеличивать ресурсы не требуется, но в обозримом будущем это необходимо будет сделать. Возможно, стоит увеличить ресурсы сейчас, чтобы предотвратить их недостаток в будущем
- change - необходимо увеличить ресурсы диска на чтение. Возможно, стоит купить диск с большей скоростью чтения, заменить hdd на ssd, заменить сетевой диск на локальный, реализовать RAID-массив или вынести БД (или выполнить шардирование БД, вынеся части БД на разные серверы)

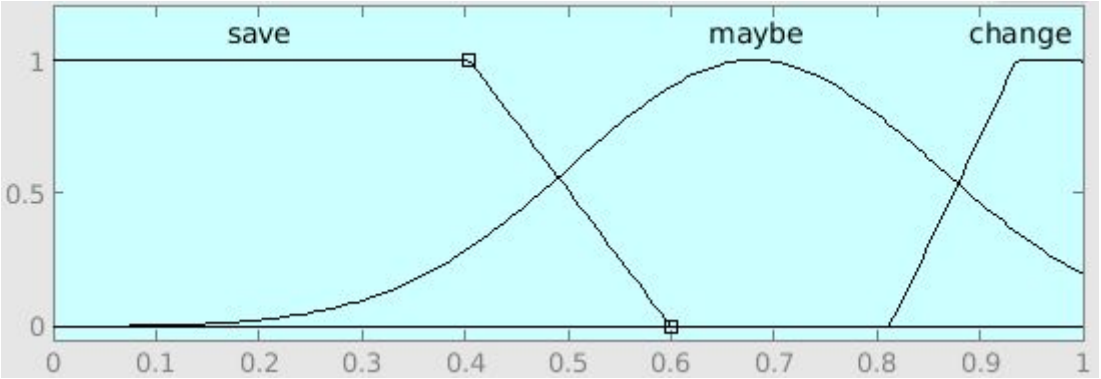


4. disk_write

Показывает, стоит ли увеличивать ресурсы диска на запись. Терм-множество:

- save - увеличивать ресурсы не нужно

- maybe - пока увеличивать ресурсы не требуется, но в обозримом будущем это необходимо будет сделать. Возможно, стоит увеличить ресурсы сейчас, чтобы предотвратить их недостаток в будущем
- change - необходимо увеличить ресурсы диска на запись. Возможно, стоит купить диск с большей скоростью записи или вынести БД на отдельный сервер (если это уже сделано, то нужно уменьшить сетевую латентность между сервером веб-приложения и сервером БД)



Построение базы нечетких лингвистических правил

Определим правила для системы в индексной форме:

```
1 0 0 0 0 0, 3 0 0 0 (1) : 1
3 0 0 0 0 1, 3 0 0 0 (1) : 1
3 0 0 0 0 2, 3 0 0 0 (0.4) : 1
3 0 0 0 0 4, 2 0 0 0 (0.75) : 1
3 0 0 0 0 3, 1 0 0 0 (1) : 1
2 0 0 0 0 0, 1 0 0 0 (1) : 1
0 3 -1 0 0 0, 0 3 0 0 (1) : 1
0 3 1 0 0 0, 0 2 0 0 (1) : 1
0 3 3 0 0 0, 0 3 0 0 (0.5) : 1
0 2 -2 0 0 0, 0 1 0 0 (1) : 1
0 2 2 0 0 1, 0 3 0 0 (1) : 1
0 2 2 0 0 2, 0 3 0 0 (0.4) : 1
0 2 2 0 0 4, 0 2 0 0 (0.75) : 1
0 2 2 0 0 3, 0 1 0 0 (1) : 1
0 1 0 0 0 0, 0 1 0 0 (1) : 1
0 0 0 0 1 0, 0 0 1 0 (1) : 1
0 0 0 0 3 0, 0 0 0 1 (1) : 1
0 0 0 1 0 0, 0 0 1 1 (1) : 1
0 0 0 3 -3 0, 0 0 0 3 (1) : 1
0 0 0 3 -1 0, 0 0 3 0 (1) : 1
0 0 0 2 -3 1, 0 0 0 3 (1) : 1
0 0 0 2 -1 1, 0 0 3 0 (1) : 1
0 0 0 2 -3 2, 0 0 0 3 (0.4) : 1
0 0 0 2 -1 2, 0 0 3 0 (0.4) : 1
0 0 0 2 -3 4, 0 0 0 2 (0.75) : 1
0 0 0 2 -1 4, 0 0 2 0 (0.75) : 1
0 0 0 2 -3 3, 0 0 0 1 (1) : 1
0 0 0 2 -1 3, 0 0 1 0 (1) : 1
```

В форме естественных языков (Рис. 1, Рис. 2, Рис. 3):

1. If (cpu_usage is buzy) then (cpu is change) (1)
2. If (cpu_usage is middle) and (la_value is very_high) then (cpu is change) (1)
3. If (cpu_usage is middle) and (la_value is high) then (cpu is change) (0.4)
4. If (cpu_usage is middle) and (la_value is middle) then (cpu is maybe) (0.75)
5. If (cpu_usage is middle) and (la_value is low) then (cpu is save) (1)
6. If (cpu_usage is free) then (cpu is save) (1)

Рис. 1 (cpu_rules)

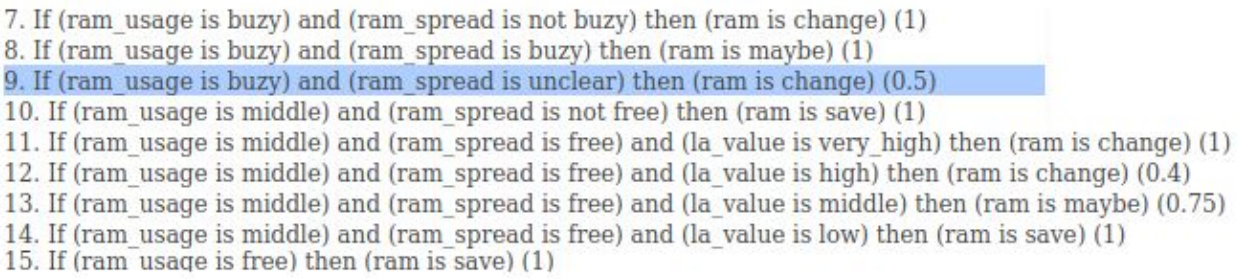


Рис. 2 (ram_rules)

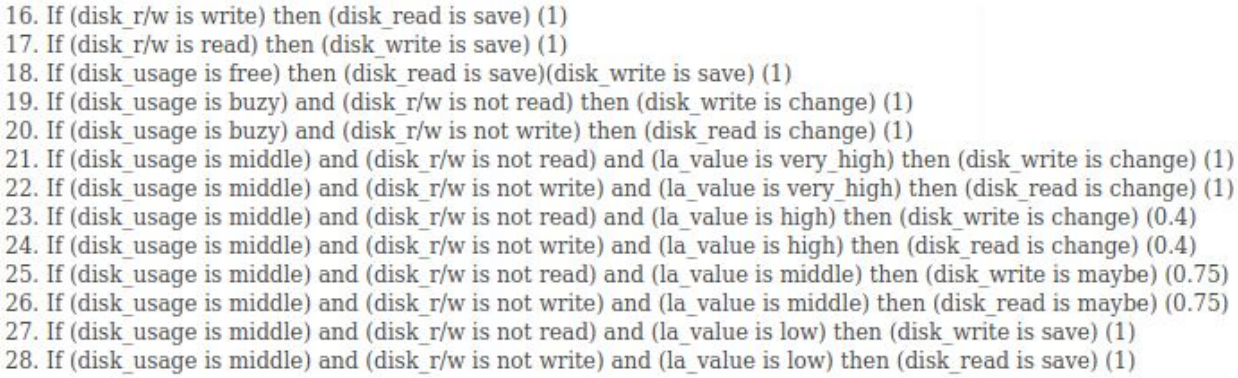


Рис. 3 (disk_rules)

Вероятно, представленные эмпирические правила достаточно просты с точки зрения понимания принципов, на которых они были построены. Стоит обратить внимание на то, что la_value используется исключительно в посылках со значением middle измеряемого соответствующего аппаратного ресурса. la_value необходимо для того, чтобы определить величину потребности в аппаратных ресурсах. Чем больше la_value, тем вероятнее, что аппаратный ресурс будет необходим в ближайшее время (значение maybe) или то, что он необходим уже сейчас (значение change).

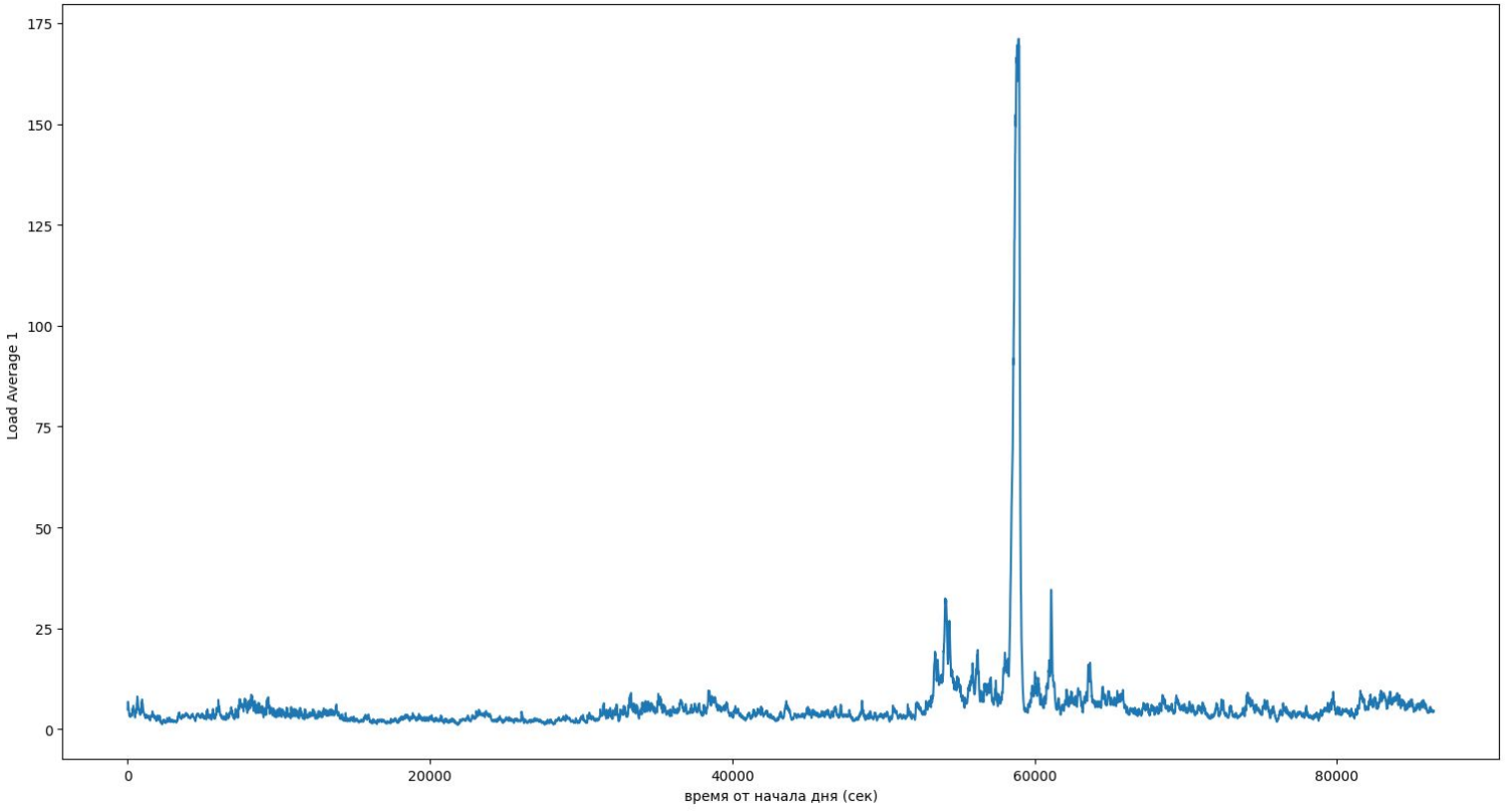
Код проекта в формате .fis можно найти в github.

Анализ системы

Рассмотрим пример работы разработанной системы на реальном примере для иллюстрации расчетов входных переменных и общей схемы использования разработанной системы.

Для анализа будем использовать лог утилиты atop (пожалуй, самая функциональная из top-утилит), которая делала снапшот каждые 10 секунд начиная с 00:00:05.

Построим график Load Average 1 (средний Load Average за последнюю минуту) рассматриваемого сервера за интересующую нас дату (08.06.2020):

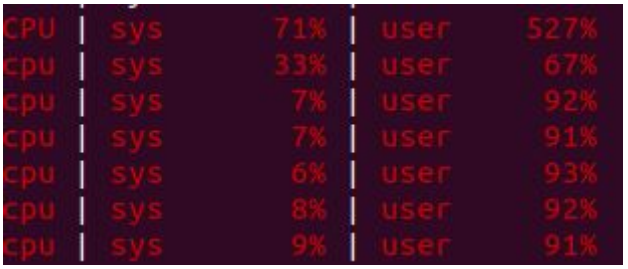


Как видно, в районе 16:00 (16:11 - 16:25) наблюдается резкий рост Load Average. Максимальный Load Average (171.1) наблюдался в 16:22:26 (время сместилось на 1 секунду, так как утилита atop отмеряет 10 секунд с момента окончания создания снапшота, то есть каждый раз теряется время, равное скорости создания снапшота).

Вычислим входные переменные для разработанной системы:

- cpu_usage

  В 16:22:26 процент использования CPU равен 598 (из 600, так как в CPU 6 ядер).



  Соответственно, значение параметра: 598 / 600 = 0.9966666666666667

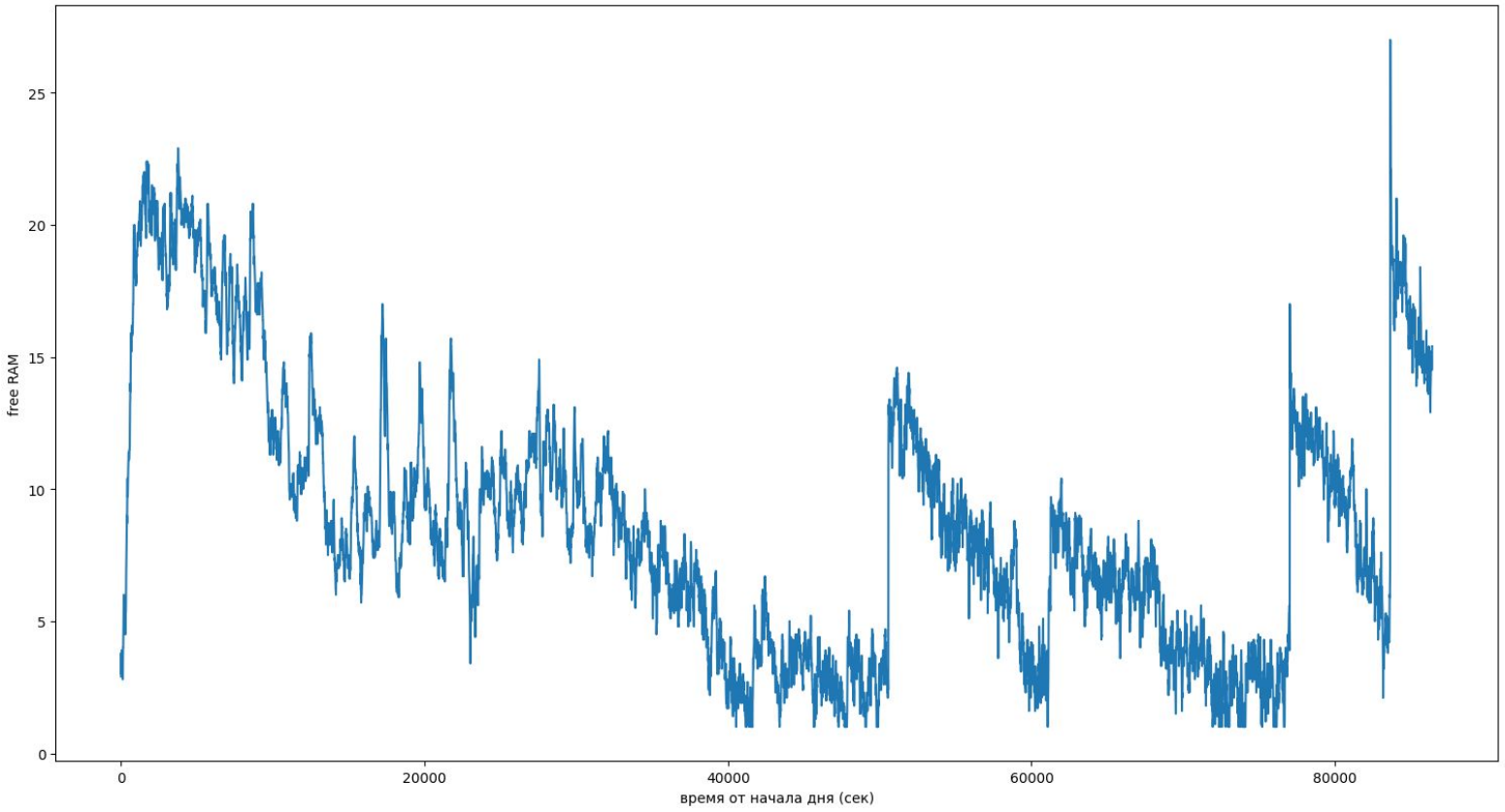
- ram_usage

  В 16:22:26 свободно 8.2 Гб.

  Значение параметра: (62.8 - 8.2) / 62.8 = 0.8694267515923566

- ram_spread

  Рассмотрим график свободной RAM за интересующую нас дату:



Сопоставив график Load Average и данный график RAM, можно увидеть, что среднее значение RAM за остальной день (больше Load Average не росло так резко как в момент рассматриваемого периода) составляет приблизительно 12 Гб.

Соответственно, значение параметра: 12.0 / 62.8 = 0.1910828025477707

- disk_usage

  В 16:22:26 использовалось 2% ресурсов диска.

  ```
  DSK |          sda | busy      2%
  ```

  Значение параметра: 0.02

- disk_r/w

  Посмотрим, для чтения или для записи используется диск:

  ```
  read       0 | write   1794
  ```

  Соответственно, значение параметра: 0.0

- la_level

  Значение Load Average 1 в 16:22:26: 171.1

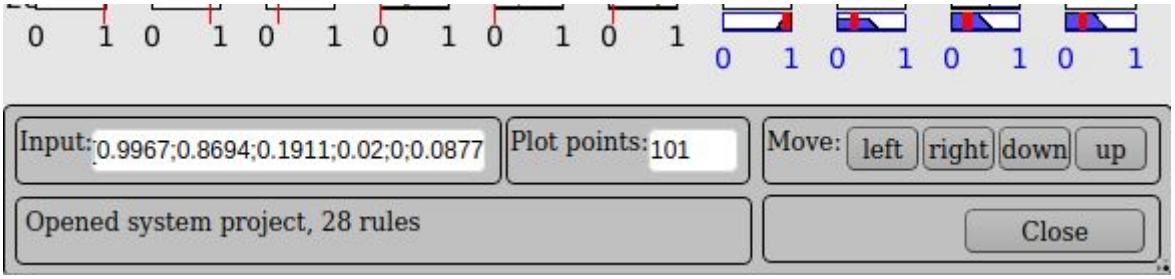
  ```
  CPL | avg1  171.10
  ```

  До начала рассматриваемого периода резкого роста Load Average равнялся приблизительно 15.0

  Тогда значение параметра: 0.08766803039158387

Код, используемый для построения представленных графиков можно найти в github. Код используемый для подготовки и парсинга лога atop также можно найти в github.

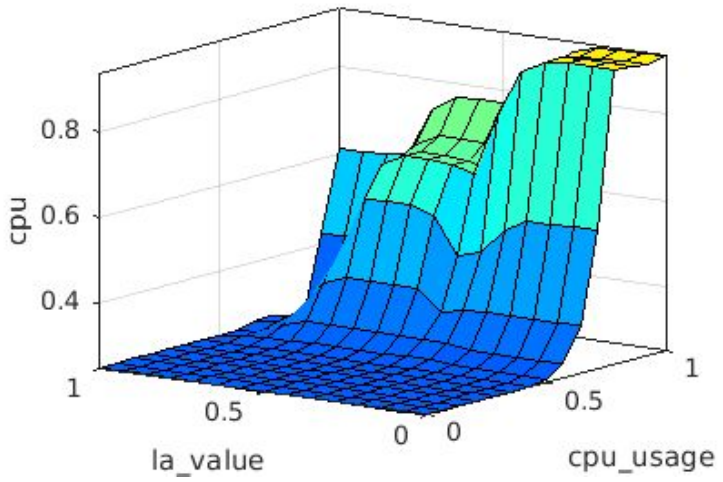
Видно, что в данной ситуации необходимо увеличивать ресурсы CPU. Посмотрим, как разработанная система отработает на данных входных параметрах:



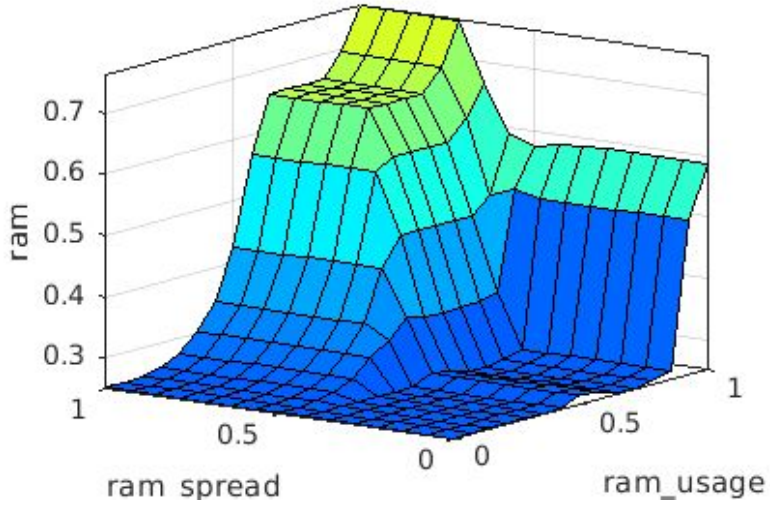
В правом верхнем углу скриншота видно, что на первом графике (выходная переменная cpu) система отметила необходимость увеличения аппаратных ресурсов CPU, при этом для других аппаратных ресурсов отобразив значение save. Данный результат корректен.

Графики поверхностей

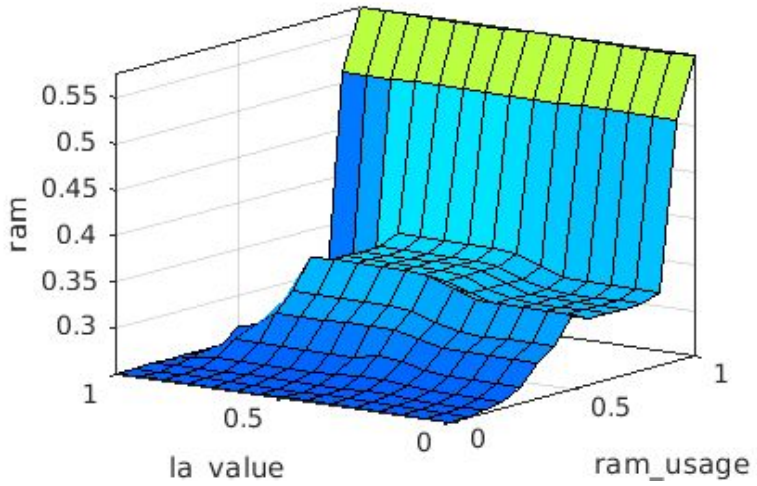
- Зависимость cpu_usage - la_value - cpu



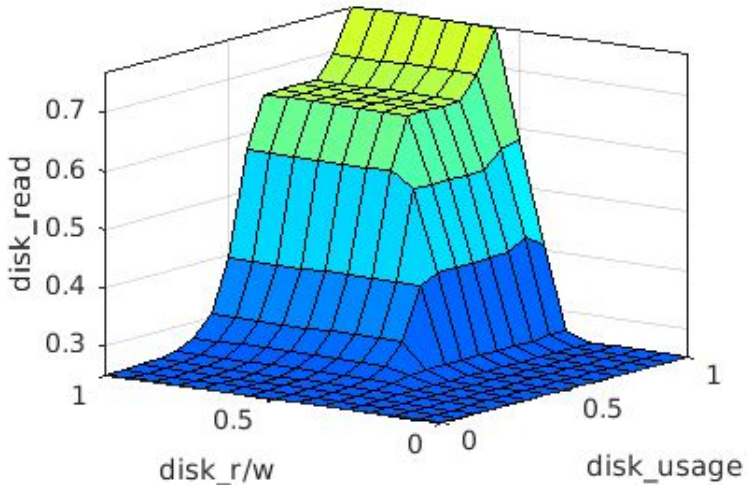
- Зависимость ram_usage - ram_spread - ram



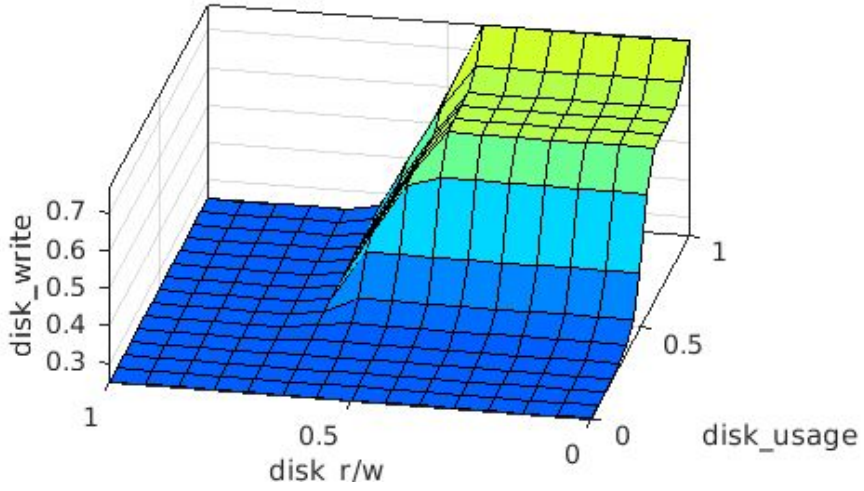
- Зависимость ram_usage - la_value - ram



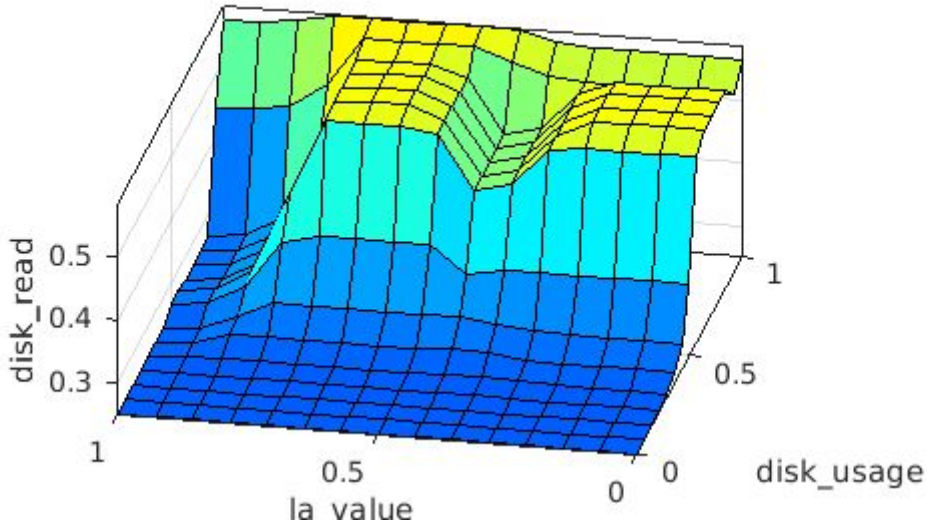
- Зависимость disk_usage - disk_r/w - disk_read



- Зависимость disk_usage - disk_r/w - disk_write



- Зависимость disk_usage - la_value - disk_read (disk_write аналогично)



Заключение

Полученная система работает корректно в рамках заложенной в нее функциональности. В примере из раздела “Анализ системы” проблема заключалась не только в недостатке CPU. Такой резкий рост Load Average был вызван также некорректно указанным лимитом сервера БД на количество одновременных соединений. Этого система, конечно же, не смогла определить и увеличение мощности CPU не помогло бы избавиться от проблем с Load Average. Тем не менее, возложенную на систему задачу она выполнила.

Также хочется отметить, что данная система могла бы использоваться для полностью автоматического увеличения аппаратных ресурсов на облачном хостинге (VPC и т.п.) “на лету”. Конечно, это потребовало бы сильного усложнения системы и, возможно, комбинирования с методами машинного обучения, но в итоге позволило бы в автоматическом режиме масштабировать ресурсы виртуальных машин, позволяя избежать проблем с временной недоступностью веб-приложений.

Еще одним сценарием использовании системы с подобным функционалом могло бы быть создание рекомендательной системы для пользователей услуг хостинга с фиксированной конфигурацией аппаратного обеспечения (в рамках услуг выделенного сервера, VPS и т.п.) для оценки необходимости увеличения аппаратных ресурсов на текущем сервере. С учетом фиксированных аппаратных ресурсов можно было бы построить систему с высоким уровнем точности рекомендаций, упростив работу владельцам веб-приложений или их разработчикам, которые зачастую не имеют нужных компетенций в области системного администрирования для самостоятельного выявления “узких мест” в текущей конфигурации.

Список литературы

1. Д. К. Потапов: “Неклассические логики. Учебное пособие.” - Изд. Санкт-Петербургский университет. 2006.
2. А. Леоненков: “Нечеткое моделирование в среде Matlab и FuzzyTech” - Изд. Санкт-Петербург. “БХВ Петербург” 2005.