Министерство цифрового развития, связи и массовых коммуникаций

Ордена трудового красного знамени

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский технический университет связи и информатики»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра «Бизнес-информатика»

КУРСОВАЯ РАБОТА

по дисциплине

«Теория систем и системный анализ»

на тему

«Многокритериальная оптимизация размещения виртуальных машин по физическим серверам в облачных центрах обработки данных»

Вариант № 2

Выполнил: студент первого курса

Ахтямов Булат

группы БЭИ 2102

Принял: к.т.н., доц. Тутова Н.В.

Москва 2021 г.

**СОДЕРЖАНИЕ**

Введение............................................................................................................................3

1. Исходные данные .....................................................................................................4
2. Постановка задачи.....................................................................................................5 2.1.Содержательная постановка ..............................................................................7

2.2. Математическая постановка..............................................................................8

1. Метод решения ....................................................................................................10
2. Вычислительный эксперимент................................................................................12

Заключение.......................................................................................................................15

Список источников и литературы .................................................................................16

Приложение. Программный код....................................................................................17

**ВВЕДЕНИЕ**

С каждым годом требование заказчиков и ожидания потребителей переходят на новые уровни, возникает нужда в улучшении оборудования и оптимизации кодов. Причём нельзя обойтись только одним, так как при хорошем оборудовании, но плохой оптимизации работа компьютерных машин будет не оптимальна и энергозатратна.

Центры обработки данных должны предоставлять достаточное количество ресурсов для бесперебойной работы размещенных в них приложений, нагрузка на которых может значительно меняться во времени. В случаях, когда эти изменения выходят за рамки допустимых значений, работоспособность приложений снижается, что требует динамического перераспределения ресурсов. В облачных ЦОД распределение ресурсов осуществляется путем миграции виртуальных машин (ВМ) – перемещения виртуальных машин между физическими серверами. Если миграция происходит без отключения и потери доступности ВМ, то такая миграция носит название “живой”. Такая миграция позволяет выполнять соглашения об уровне сервиса (SLA), сбалансировать нагрузку между физическими серверами ЦОД, а также размещать виртуальные машины на меньшем числе физических серверов.

Кроме этого, появились сеть центров обработки данных и информационные ресурсы могут перераспределяться между ЦОД в соответствии со спросом на ресурсы, например, с учётом часовых поясов. Это заставляет искать эффективные и быстрые алгоритмы распределения ресурсов с учетом растущих размерностей задач.

Процесс динамического распределения ресурсов включает в себя три этапа: мониторинг серверов и обнаружение критических ситуаций, выбор виртуальных машин и выбор серверов назначения.  
В данной работе основное внимание уделяется третьему этапу выбора серверов для размещения виртуальных машин. Поставлена задача многокритериальной оптимизации, выбран метод ее решения.

**ИСХОДНЫЕ ДАННЫЕ**

**Заданные характеристики сервера** (, ) = 6

**Производительность процессора** () = {0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3}

**Объем оперативной памяти** () = {0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3}

**Количество серверов** (*N*) = 5

**Задействованная часть ресурсов** ( , ) = {0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3}

**2. Постановка Задачи**

Практическая часть курсовой работы посвящена решению задачи многокритериальной оптимизации размещения виртуальных машин по физическим серверам в облачных центрах обработки данных.

## Архитектура системы

Динамическое размещение вычислительных ресурсов с включением процесса живой миграции виртуальных машин является основным этапов цикла системы управления облачным центром обработки данных, структурная схема которой приведена на рисунке 1.



**Рис. 1.** Двухуровневая система управления ресурсами облачного ЦОД

Типовая система управления ресурсами облачного ЦОД имеет двухуровневую архитектуру, состоящую из глобального и локальных контроллеров. На локальных контроллерах постоянно анализируются данные системы мониторинга о состоянии физических серверов, на которых они расположены. Определяются возможные состояния недогрузки, перегрузки и перегрева на основании прогноза для следующего окна наблюдения. Проверка показателей системы осуществляется последовательно в соответствии с важностью критериев (перегрев, перегрузка, недогрузка сервера), при этом процесс наблюдения осуществляется непрерывно, в том числе в случае выполнения действий по перемещению ВМ (рис. 2).



Рис. 2

В случае выявления одного из перечисленных состояний, локальный контроллер посылает сообщение глобальному контроллеру, на котором инициируется процесс миграции виртуальных машин: выбираются виртуальные машины для миграции и серверы назначения. Выбор сервера назначения является ответственным этапом, поскольку неудачный выбор сервера может привести к новым нежелательным миграциям, поскольку сами миграции дополнительно нагружают систему и приводят к ухудшению производительности и простою виртуальных машин.

**2.1 Содержательная постановка**

Пусть имеется *N* активных физических серверов и такое же количество виртуальных машин для размещения путем миграции. Данные серверы являются работающими и могут обслуживать другие виртуальные машины.

Ресурсы *i*-ой, *i*=1,…,*N*, виртуальной машины зададим

производительностью её процессора *VMiCPU* и объемом оперативной памяти *VMiRAM*. Такие же характеристики будем считать заданными для каждого сервера *PMjCPU*\_0 и *PMjRAM*\_0. Предположим, что в рассматриваемый момент времени у каждого сервера задействована часть ресурсов другими виртуальными машинами, которую обозначим для процессора *PMjCPU*\_1 и *PMjRAM*\_1 для памяти. Предположим также, что свободной части ресурсов достаточно для размещения любой ВМ из *N* имеющихся.

Учтем цикличность процесса динамического размещения виртуальных машин, что дает возможность предположить: за один цикл работы контроллера на *i*-ый сервер одновременно может быть размещено не более одной ВМ, и *i*-ая ВМ может быть размещена только на одном, а не на двух и более серверах (рис. 3).



Рис. 3. Иллюстрация задачи размещения виртуальных машин по серверам

Требуется найти оптимальное размещение виртуальных машин с учетом минимального времени принятия решения по процессу размещения, которое не превышает цикл работы контроллера 2-5 минут.

**Критерии оптимизации**

Предположим, что размещаемая на сервере виртуальная машина занимает всю выделенную ей память и процессорное время. Тогда обозначим *uijCPU* загрузку процессора сервера *j* после размещения виртуальной машины *i*, а *uijRAM* загрузку памяти сервера *j* после размещения виртуальной машины *i.*

** (1)

** (2)

**Критерий неэффективности использования ресурсов** отражаетнасколько несбалансированно используются ресурсы на каждом сервере. Остаток ресурсов на каждом сервере должен быть сбалансирован по нескольким видам ресурсов. Например, неудачным размещением виртуальных машин является такое размещение, при котором вся память сервера использована, а процессор малозагружен.

Критерий неэффективности использования ресурсов для сервера можно сформулировать следующим образом:

 (3)

Данный критерий отражает насколько полно загружены ресурсы серверов различных типов. Значения данного критерия лежат в диапазоне от 0 до 1. Чем ближе значение критерия к нулю, тем лучше загружены ресурсы сервера.

Наиболее часто требование на качество обслуживания облачного сервиса задаётся в виде среднего времени отклика, которое зависит от загрузки процессора. При превышении некоторого порога загрузки происходит снижение производительности приложения и увеличение времени отклика. В литературе используются пороговые значения загрузки в диапазоне 80-90%. В качестве критерия **нарушения SLA-соглашений** выбрана следующая логистическая функция.

, (4)

Значение этой функции также лежат в диапазоне от 0 до 1, в точке порогового значения *uijCPU* = 80% значение функции равно 0,5 и быстро увеличивается при превышении порогового значения. Данный критерий должен быть минимизирован.

## 2.2 Математическая постановка задачи.

Введем переменные задачи *xij*, которые будут соответствовать назначению ВМ на сервер. В этом случае *xij* = 1, если *i*-ая ВМ назначается на выполнение *j*-му серверу, и *xij* = 0, если *i*-ая ВМ не назначается на выполнение *j*-му серверу.

Поставленная задача по структуре, необходимым условиям, характеру переменных эквивалентна известной основной задаче о назначениях, математическая постановка которой для представленной предметной области имеет вид:

,

,

где множество допустимых альтернатив  формируется следующей системой ограничений:



Критерии задачи относится к классу линейных. Коэффициенты критериальной функции являются матрицами размером *N* × *N*, их расчет осуществляется по формулам (1-4). Умножение коэффициентов на матрицу размещения *X* влияет в конечном итоге на значения критериев.

**3. Метод решения**

Известно, что задача о назначении может быть сведена к замкнутой транспортной задаче путем замены ограничения  на *xij* ≥ 0. Это даёт возможность использовать для получения оптимального решения известное и доступное программное обеспечение. Выберем такой метод решения задачи размещения виртуальных машин по физическим серверам.

Сведение задачи к транспортной позволяет также реализовать её многокритериальность, путем использования такого подхода как свёртка критериев со всеми её преимуществами.

В итоге постановка задачи будет выглядеть следующим образом:



при ограничениях

,

где *α*1, *α*2 – веса критериев; .

**4. Вычислительные эксперименты**

Производительность процессора виртуальных машин в ГГц равномерно распределена из следующего набора значений {0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3}, а память из набора значений {0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3}. Число допустимых серверов и ВМ будет равно 6. Вычислительный эксперимент я буду проводить на языке программирования R.  
Подключаем нужную нам библиотеку.  
library(clue)  
Записываем все данные.  
start.time <- Sys.time()

#исходные данные

#задаем число ВМ и серверов

k = 5

#задаем диапазон данных для задания значений процессора и памяти

s<-c(0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3)

samp<-sample(1:6, k, replace = TRUE)

cpu<-s[samp]

#cpu

samp<-sample(1:6, k, replace = TRUE)

ram<-s[samp]

#ram

samp<-sample(1:6, k, replace = TRUE)

cpu\_1<-s[samp]

#cpu\_1

samp<-sample(1:6, k, replace = TRUE)

ram\_1<-s[samp]

#ram\_1

#создаем виртуальные машины и хосты

vm<-data.frame(cpu,ram)

host<-data.frame(cpu\_1,ram\_1,cpu\_0=rep(6, k),ram\_0=rep(6, k))

#считаем загрузку процессора и памяти серверов

u\_cpu<-matrix(nrow=k,ncol=k)

u\_ram<-matrix(nrow=k,ncol=k)

for (i in 1:k) {

u\_cpu[i,]=(vm$cpu[i]+host$cpu\_1)/host$cpu\_0

u\_ram[i,]=(vm$ram[i]+host$ram\_1)/host$ram\_0

}

for (i in 1:k) {

for (j in 1:k){

if (u\_cpu[i,j]>1)

u\_cpu[i,j]=NaN

if (u\_ram[i,j]>1)

u\_ram[i,j]=NaN

}

}

#вычисляем значения коэффициентов целевых функций

C1<-1-u\_cpu\*u\_ram

C2<-1/(1+exp(-(u\_cpu-0.8)))

**Решим задачу при помощи Венгерского метода**

#вычисляем значения коэффициентов обобщенного критерия

for (i in 1:k) {

for (j in 1:k){

if (is.nan(C3[i,j]))

C3[i,j]=k

}

}

# используем венгерский метод

y<-solve\_LSAP(C3, maximum = FALSE)

y

cat("Не хватает ресурсов виртуальным машинам:", sum(C3[cbind(seq\_along(y), y)])%/%k)

end.time <- Sys.time()

time.taken <- end.time - start.time

time.taken

При выборке исходных данных, которые я написал выше, мой вычислительный эксперимент дал такие показания:

1) Не хватает ресурсов виртуальным машинам 0, значит всё хорошо

2) Время за которое выполнился код 0.4704959 secs

3) Оптимальное решение 1 => 4, 2 => 1, 3 => 2, 4 => 5, 5 => 3

**Решим данную задачу Симплекс методом**

Импортируем библиотеку «1pSlove».

После того, как мы вычислили значение коэффициентов, создаем матрицу коэффициентов левой части системы ограничений:

#создаем матрицу коэффициентов левой части системы ограничений

A <- matrix(0, nrow=2\*k, ncol=k\*k)

a=1

b=k

for(i in 1:k)

{

for(j in a:b)

{ A[i,j]=1}

a=b+1

b=b+k

}

a=1

for(i in (k+1):(2\*k))

{

for(j in seq(a,k\*k,k))

{A[i,j]=1}

a=a+1

}

#создаем вектор правых частей системы ограничений

B <- rep(1,2\*k)

#создаем вектор равенств

constranints\_direction <- rep("=", 2\*k)

C3<-0.5\*C1+0.5\*C2

optimum <- lp(direction="min",

objective.in = C3,

const.mat = A,

const.dir = constranints\_direction,

const.rhs = B,

all.int = T)

# Вывод статуса: 0 = успешно, 2 = нет решения

print(optimum$status)

# Отображение оптимальных значений

best\_sol <- optimum$solution

print(paste("Значение целевой функции: ", optimum$objval, sep=""))

sum(C1\*best\_sol)

sum(C2\*best\_sol)

rm(optimum, best\_sol)

end.time <- Sys.time()

time.taken <- end.time - start.time

time.taken

При использовании симплекс метода вывелось следующее:

1. Вывело «0», значит код исправен и всё работает
2. Значение целевой функции равно 3.216970759735
3. Время, за которое выполнился код 0.5694671 secs
4. Оптимальное решение 1 => 4, 2 => 1, 3 => 2, 4 => 5, 5 => 3

При сравнении этих методов я пришёл к выводу, что венгерский метод быстрее почти на 20%, поэтому при использовании больших объёмах данных лучше пользоваться Венгерским методом.

**Заключение**

Сведение основной задачи о назначении к замкнутой транспортной задаче позволило в условиях жесткого реального времени решить при многих критериях задачу размещения виртуальных машин и значительно увеличить ее размерность в сравнении с эвристическими алгоритмами, что дает возможность поддерживать качество современных облачных сервисов в условиях стремительного роста физических и виртуальных ресурсов центров обработки данных.

Разработанная математическая постановка задачи, результат вычислительного эксперимента могут быть включены в математическое обеспечение процессов живой миграции виртуальных машин.

**Список литературы**

1. Воронцов А.С, Тутова Н.В., Тутов А.В. Динамическое распределение вычислительных ресурсов центров обработки данных //T-Comm: Телекоммуникации и Транспорт. 2016. Т. 10. №7.  
   2. Воронцов А.С, Тутова Н.В., Тутов А.В. Оптимизация размещения облачных серверов в центрах обработки данных //T-Comm: Телекоммуникации и транспорт. №6. 2015.  
   3. Микрюков А.А., Хантимиров Р.И. Задача первоначального выделения ресурсов в облачных вычислительных средах на основе метода анализа иерархий // Статистика и Экономика. 2015. №4. С. 184-187с  
   4. Тутов А.В. Модели и методы распределения ресурсов инфокоммуникационный системы облачных центров обработки данных // Наукоемкие технологии в космических исследованиях Земли. 2018. Т.10.№6.  
   5. Ногин В.Д Линейная свертка критериев в многокритериальной оптимизации // Искусственный интеллект и принятие решений. 2014. №4. С. 73-82.

**Приложение. Программный код**

**Венгерский метод**

library(clue)

start.time <- Sys.time()

k=5

s<-c(0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75,3)

samp<-sample(1:6, k, replace = TRUE)

cpu<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

ram<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

cpu\_1<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

ram\_1<-s[samp]

vm<-data.frame(cpu,ram)

host<-data.frame(cpu\_1,ram\_1,cpu\_0=rep(6,k),ram\_0=rep(6,k))

u\_cpu<-matrix(nrow=k,ncol=k)

u\_ram<-matrix(nrow=k,ncol=k)

for (i in 1:k) {

u\_cpu[i,]=(vm$cpu[i]+host$cpu\_1)/host$cpu\_0

u\_ram[i,]=(vm$ram[i]+host$ram\_1)/host$ram\_0

}

for (i in 1:k) {

for (j in 1:k){

if (u\_cpu[i,j]>1)

u\_cpu[i,j]=NaN

if (u\_ram[i,j]>1)

u\_ram[i,j]=NaN

}

}

C1<-1-u\_cpu\*u\_ram

C2<-1/(1+exp(-(u\_cpu-0.8)))

C3<-C1 \* 0.5 + C2 \* 0.5

for (i in 1:k) {

for (j in 1:k){

if (is.nan(C3[i,j]))

C3[i,j]=k

}

}

y<-solve\_LSAP(C3, maximum = FALSE)

y

cat("Не хватает ресурсов виртуальным машинам:", sum(C3[cbind(seq\_along(y), y)])%/%k)

end.time <- Sys.time()

time.taken <- end.time - start.time

time.taken

**Симплекс**

library(lpSolve)

start.time <- Sys.time()

k = 5

s<-c(0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3)

samp<-sample(1:6, k, replace = TRUE)

cpu<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

ram<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

cpu\_1<-s[samp]

samp<-sample(1:6, k, replace = TRUE)

ram\_1<-s[samp]

vm<-data.frame(cpu,ram)

host<-data.frame(cpu\_1,ram\_1,cpu\_0=rep(6,k),ram\_0=rep(6,k))

u\_cpu<-matrix(nrow=k,ncol=k)

u\_ram<-matrix(nrow=k,ncol=k)

for (i in 1:k) {

u\_cpu[i,]=(vm$cpu[i]+host$cpu\_1)/host$cpu\_0

u\_ram[i,]=(vm$ram[i]+host$ram\_1)/host$ram\_0

}

for (i in 1:k) {

for (j in 1:k){

if (u\_cpu[i,j]>1)

u\_cpu[i,j]=NaN

if (u\_ram[i,j]>1)

u\_ram[i,j]=NaN

}

}

C1<-1-u\_cpu\*u\_ram

C2<-1/(1+exp(-(u\_cpu-0.8)))

A <- matrix(0, nrow=2\*k, ncol=k\*k)

a=1

b=k

for(i in 1:k)

{

for(j in a:b)

{ A[i,j]=1}

a=b+1

b=b+k

}

a=1

for(i in (k+1):(2\*k))

{

for(j in seq(a,k\*k,k))

{A[i,j]=1}

a=a+1

}

B <- rep(1,2\*k)

constranints\_direction <- rep("=", 2\*k)

C3<-0.5\*C1+0.5\*C2

optimum <- lp(direction="min",

objective.in = C3,

const.mat = A,

const.dir = constranints\_direction,

const.rhs = B,

all.int = T)

print(optimum$status)

best\_sol <- optimum$solution

print(paste("Значение целевой функции: ", optimum$objval, sep=""))

sum(C1\*best\_sol)

sum(C2\*best\_sol)

rm(optimum, best\_sol)

end.time <- Sys.time()

time.taken <- end.time - start.time

time.taken