УДК 004.75

ДИНАМИЧЕСКИЙ МЕТОД ОЦЕНКИ ЗАГРУЗКИ УЗЛОВ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

Т.А. РАДИВИЛОВА, И.Н. ИВАНИСЕНКО
Харьковский национальный университет радиоэлектроники

Abstract *– This paper provides a topical scientific problem solution related to the load balancing and efficient utilization of resources of the distributed system. The paper presents a method of estimating resources of the distributed system, such as network nodes, the processor, memory, and bandwidth. The proposed method allows to calculate the loading of each node separately in a distributed system and the entire system. Classes of service flows taken into account in the calculation of these resources loading. The complex value of imbalance of load server entered, which taking into account the weight coefficients for processor, memory, and network bandwidth. These weight coefficients allow to select the importance of each network resource (CPU, memory and bandwidth) compared with each other. Also, this method allows to calculate the imbalance of the system servers. Using the method in load balancing allows to distribute requests by servers such way that deviation of the load servers from the average value was minimal, that allow to provide higher system performance parameters (utilization efficiency) and faster processing flows.*

Анотація *– В роботі запропоновано вирішення актуальної наукової задачі, яка пов'язана з балансуванням навантаження і ефективністю використання ресурсів розподіленої системи. Запропонований метод базується на розрахунку завантаження процесора, пам'яті і пропускної здатності каналу потоками різних класів обслуговування для кожного сервера і всієї розподіленої системи. Використання запропонованого методу при балансуванні навантаження дозволяє розподіляти запити по серверам так, щоб відхилення завантаженості серверів від середнього значення було мінімальним, що дозволить отримати більш високі показники продуктивності системи і швидшу обробку потоків.*

Аннотация *– В работе предложено решение актуальной научной задачи, связанной с балансировкой нагрузки и эффективностью использования ресурсов распределенной системы. Предложенный метод основан на расчете загрузки процессора, памяти и пропускной способности канала потоками разных классов обслуживания для каждого сервера и всей распределенной системы. Использование предложенного метода при балансировке нагрузки позволяет распределять запросы по серверам так, чтобы отклонение загруженности серверов от среднего значения было минимальным, что позволит получить более высокие показатели производительности системы и более быструю обработку потоков.*

Введение

В настоящее время вместе с планомерным увеличением скоростей передачи данных в телекоммуникациях увеличивается доля интерактивного трафика, крайне чувствительного к параметрам среды транспортировки. Для предоставления требуемого количества ресурсов при передаче различных видов трафика, предъявляющих различные требования к характеристикам телекоммуникационной сети используются различные механизмы обеспечения QoS (Quality of service). Одним из таких механизмов является балансировка нагрузки [1-3]. Система балансировки нагрузки решает задачи обеспечения качества обслуживания и повышения производительности распределенных систем за счет оптимального распределения заданий между узлами вычислительной системы. Для наиболее полного использования всех имеющихся ресурсов распределенной системы используются различные методы оценки загрузки узлов и системы в целом.

Оценка загрузки вычислительного узла может производиться несколькими способами. Один из способов аналитический, который заключается в приблизительной оценке загрузки каждого объекта на основе данных о поступивших задачах. Преимущество аналитического метода состоит в том, что он достаточно точно может оценить трудоёмкость задачи. Недостаток же этого метода состоит в том, что он может быть довольно неточным в случае, если модель для оценки скорости выполнения задачи неточна.

Другой способ сбора данных о загрузке состоит в измерении загрузки узлов. Большинство современных машин снабжено счетчиками времени (с точностью до микросекунд), которые могут быть использованы для измерения времени выполнения каждой задачи. Преимущество метода состоит в том, что он является точным в большинстве случаев. К недостаткам можно отнести следующее: стратегии балансировки, основанные на этом методе, учитывают прошлое распределение нагрузок. Если трудоемкость задач меняется непредсказуемым образом, то метод будет неточным.

Наиболее известными исследованиями в области балансировки, теоретических исследований и разработки фундаментальных основ распределения нагрузки, в создании математического аппарата, моделей и методов управления для распределения нагрузки в распределенных системах занимались такие ученые как Е.И. Игнатенко [2], В.Н. Тарасов [3], F. Wang [4], V. Cardellini [1, 5], S.Keshav [6], Xing-Guo Luo, Xing-Ming Zhang [7], Hisao Kameda, Lie Li [8], а также многие другие ученые работающие над проблемами распределения нагрузки.

Целью данной работы является разработка метода расчета загрузки узлов и дисбаланса распределенной системы, основанного на оценивании загрузки процессора, памяти и пропускной способности канала.

I. Решение задачи балансировки

Проблема балансировки вычислительной нагрузки возникает по нескольким основным причинам [1, 5, 8]:

- структура распределенного приложения неоднородна, различные логические процессы требуют различных вычислительных мощностей;
- структура распределенной системы также неоднородна, т.е. разные вычислительные узлы обладают разной производительностью;
- структура межузлового взаимодействия неоднородна, т.к. линии связи, соединяющие узлы, могут иметь различные характеристики пропускной способности.

В зависимости от поставленной задачи можно использовать статическую или динамическую балансировку [9]. Статическая балансировка выполняется до начала выполнения задач. Однако предварительное размещение логических процессов по процессорам (серверам) не даёт эффекта. Это объясняется изменчивостью вычислительной среды (узел может выйти из строя), занятостью узла другими вычисления-

ми. Так или иначе, выигрыш от распределения логических процессов по серверам с целью выполнения параллельной обработки становится неэффективным. Динамическая балансировка предусматривает распределение вычислительной нагрузки на узлы во время выполнения задач. Программное обеспечение, реализующее динамическую балансировку, определяет: загрузку вычислительных узлов; пропускную способность линий связи; количество свободной памяти; частоту обменов сообщениями между логическими процессами задач и др. На основании собранных данных о задачах и вычислительной среде принимается решение о распределении задач между узлами сети.

Цель балансировки загрузки может быть сформулирована следующим образом: исходя из набора задач, включающих вычисления и передачу данных, и сети серверов определенной топологии найти такое распределение задач по серверам, которое обеспечивает примерно равную вычислительную загрузку серверов и минимальные затраты на передачу данных.

Обычно практичное и полное решение задачи балансировки загрузки состоит из четырех этапов [3, 5, 9, 10]: 1) оценка загрузки вычислительных узлов; 2) инициация балансировки загрузки; 3) принятие решений о балансировке; 4) распределение задач.

Опишем более подробно каждый этап балансировки.

1. Оценка загрузки системы.

На этом этапе осуществляется расчет загрузки каждого сервера путем расчета среднего коэффициента использования процессора, памяти, пропускной способности сети i-го сервера. Полученная информация о загрузке используется для процесса балансировки, во-первых, для определения возникновения дисбаланса, во-вторых, для определения нового распределения задач путем вычисления объема работ, необходимого для перемещения задач. Отсюда, качество работы балансировки загрузки напрямую зависит от точности и полноты информации.

2. Инициация.

Слишком частое выполнение балансировки загрузки может привести к тому, что выполнение задач только замедлится. Затраты на саму балансировку могут превзойти возможную выгоду от ее проведения. Следовательно, для продуктивности балансировки необходимо каким-то образом определять момент ее инициализации. Для этого следует: определить момент возникновения дисбаланса загрузки; определить степень необходимости балансировки путем сравнения возможной пользы от ее проведения и затрат на нее.

Дисбаланс загрузки может определяться синхронно и асинхронно. При синхронном определении дисбаланса все процессоры (серверы сети) прерывают работу в определенные моменты синхронизации и определяют дисбаланс загрузки путем сравнения загрузки отдельного процессора с общей средней загрузкой. При асинхронном определении дисбаланса каждый сервер хранит историю своей загрузки. В этом случае момент синхронизации для определения степени дисбаланса отсутству-

ет. Вычислением объема дисбаланса занимается фоновый процесс, работающий параллельно с задачами.

3. Принятие решений в процессе балансировки.

Большинство стратегий динамической балансировки загрузки можно отнести к классу централизованных или к классу полностью распределенных.

При централизованной стратегии балансировщик собирает глобальную информацию о состоянии всей вычислительной системы и принимает решение о перемещении задач для каждого из серверов.

При полностью распределенной стратегии на каждом сервере выполняется алгоритм балансировки загрузки, обменивающийся информацией о состоянии с другими серверами. Перемещение задач происходит только между соседними процессорами.

4. Распределение задач.

Существует множество алгоритмов балансировки (распределения задач). Однако каждый из алгоритмов имеет как преимущества, так и недостатки [11; 12]. Наиболее часто используемыми алгоритмами балансировки нагрузки являются следующие: Round Robin Scheduling [6; 13; 14], Max-Min Algorithm [13, 15], Compare and Balance [11, 16] и другие. Можно выбрать наиболее подходящий алгоритм в зависимости от поставленных задач.

II. Описание системы балансировки

В каждый момент $t \in T$ на балансировщик LB поступает трафик интенсивностью $\lambda = [\lambda_1, \lambda_2, .., \lambda_\sigma]$, относящийся к qs-му классу обслуживания, который необходимо доставить на i-й сервер $Serv_i$ для обработки, не превышая заданных максимально допустимых значений задержки τ_{qs} и максимально допустимого процента потерь l_{qs} в зависимости от текущей загрузки серверов и реальной пропускной способности в конкретный момент времени.

Трафик обладает множеством характеристик $V = \{\lambda, h, \mu\}$, где $\lambda = [\lambda_1, \lambda_2, .., \lambda_\sigma]$ - потоки заявок (пакетов) различной интенсивности; $h = [H, h(q), \Delta h]$, где $h(q)$ - выборочное значение функции обобщенного показателя Херста, $H = h(2)$ - значение параметра Херста, $\Delta h = h(q_{min}) - h(q_{max})$ - диапазон значений обобщенного показателя Херста для участка трафика, μ - трудоемкость запроса. Трудоемкость запроса определяется как вектор требуемых ресурсов $\mu = (CPU, Net, RAM)$ для выполнения запроса. Каждому qs-му классу обслуживания соответствует набор векторов требуемых ресурсов $\mu_{qs} = (CPU, Net, RAM)$.

Балансировщик нагрузки и сервера соединены между собой двусторонними сетевыми связями $Link_{lk} = \{L_{lk}\}$, $lk = 1, 2, ..,$ которые делятся на k каналов с пропускной способностью $Net_{lk}^{k}(t) = \{Net_{lk}^{k}\}$ в момент времени t [Kameda, 1997; Tian, 2014]. Каждый i-й сервер характеризуется следующими параметрами: $CPU_i^{n_i}$ средняя загрузка

ЦПУ i-го сервера за период времени T, где n_i - количество ЦПУ (ядер) на i-м сервере (то, как ядра распределены по процессорам, будем считать несущественнным: два четырехядерных процессора соответствуют четырем двуядерным и соответствуют восьми одноядерным процессорам, имеет значение лишь общее число ядер); средний коэффициент использования памяти i-го сервера $RAM_i^{m_i}$ за период времени T, где m_i - количество памяти на i-м сервере; средний коэффициент использования пропускной способности канала $Net_i^{k_i}$ за период времени T, где k_i - количество каналов в линии связи между балансировщиком и i-м сервером.

На вход балансировщика загрузки LB поступают несколько независимых мультифрактальных потоков пакетов с различной интенсивностью $\lambda_1, \lambda_2, .., \lambda_\sigma$, каждый из которых отправляется в очередь Q_w ограниченной емкости. Время обслуживания заявок зависит от класса обслуживания qs, то есть учитывается приоритетность заявок (наивысший приоритет – первым). Пока все приоритетные запросы на обслуживание не будут обработаны, пакеты других типов остаются в очереди до истечения их времени жизни. Вновь поступившие приоритетные запросы прерывают обработку неприоритетных и с вероятностью равной единице вытесняют их в накопитель (если есть свободные места ожидания), либо за пределы системы (если свободных мест нет). Вытесненные с обслуживания пакеты присоединяются к очереди неприоритетных требований и могут быть дообслужены после всех приоритетных. Накопители являются раздельными для каждого входного потока, свободные места ожидания полнодоступны для любого вновь поступившего запроса.

В отличие от типовых приоритетных СМО рассматриваемая система снабжена вероятностным выталкивающим механизмом. Приоритетный пакет, заставший все места ожидания занятыми в момент обработки другого приоритетного пакета, с заданной вероятностью вытесняет из накопителя один из менее приоритетных пакетов и занимает его место. Вытесненный пакет теряется либо отправляется обратно в очередь. Подсистема балансировки загрузки LB в соответствии с заложенным в нее алгоритмом осуществляет извлечение задач из очередей Q_w и назначение их на свободные вычислительные ядра подходящих серверов.

Для описания механизма высвобождения занятых трафиком сетевых ресурсов при окончании передачи трафика qs-го класса обслуживания (это происходит на основе данных, поступающих от протокола маршрутизации, поддерживающего сообщения о доступной полосе пропускания и доступных ресурсах на сервере, например, CSPF, SNMP), введем переменную LB $\varepsilon_{Net_{lk}}^{qs,t_0}(t) = \{0,1\}$, указывающую, что в момент t на сервер перестал поступать трафик класса $qs(\varepsilon = 1)$, который был принят на обслуживание в момент t_0 и должен был передаваться по пути $Net_i^{k_i}(t)$ на сервер $Serv_i$. Данная переменная содержит все необходимые данные для определения сетевых ресурсов, подлежащих высвобождению.

Балансировщик LB в t-й момент характеризуется коэффициентом потерь $X_{LB}^{qs}(t) \in X$, средним временем ожидания пакета в очереди $T_{LB}^{qs}(t) \in T$.

Переменная $X_{LB}^{qs}(t) \in X$ равна проценту потерь на балансировщике трафика с классом обслуживания qs, передаваемого по пути $Net_{lk}^{k}(t)$ на сервер $Serv_i$ в момент t. Предполагается, что вероятностью искажения пакета в тракте можно пренебречь и потери происходят исключительно в балансировщике из-за переполнения буфера.

III. Анализ состояния распределенной системы

Мониторинг состояния серверов и свободной пропускной способности [2; 3, 17] можно осуществить тремя способами:

- после каждого поступившего запроса;
- в фиксированные промежутки времени, определяемые статическим алгоритмом;
- в нефиксированные промежутки времени, определяемые динамическим алгоритмом.

Информация, полученная первым способом, является наибольшей по объему, т.к. измерения проводятся после каждого поступившего запроса. При втором способе количество информации постоянно, но необходимо определить интервал съема информации, чтобы объем информации не был избыточным и недостаточным. При третьем способе количество информации зависит от частоты интервалов контроля, который должен приспосабливаться к структуре поступающего трафика, учитывая его самоподобную структуру.

Перед тем как описать стратегию балансировки нагрузки, которая включает в себя комплексное измерение общего уровня дисбаланса системы (степени равномерности распределения нагрузки между серверами), необходимо ввести некоторые понятия и определения.

Под термином "доля" будем понимать часть ресурсов процессора, выделенных для задачи. Если задаче выделяется больше долей процесса, чем другим задачам, такая задача получает больше процессорных ресурсов от планировщика долевого распределения. Доли процессора не эквивалентны процентам ресурсов процессора. Доли позволяют определить важность рабочих нагрузок по отношению к другим рабочим нагрузкам. При назначении задаче долей процессора наиболее важным является не количество долей, выделенных задаче. Гораздо важнее знать, сколько долей выделено задаче по сравнению с другими задачами. Следует также учитывать, что многие из этих других задач будут конкурировать с данной задачей за ресурсы процессора.

Загрузку процессора, памяти и канала будем рассматривать как безразмерные величины, значения которых нормированы и лежат в диапазоне [0, 1]. Общепризнано, что если среднее значение загрузки постоянно превышает 0.70, следует выяснить причину такого поведения системы во избежание проблем в будущем; если же сред-

няя загрузка системы близка к единице, то необходимо срочно найти причину и устранить ее.

IV. Комплексное измерение дисбаланса системы

Учитывая преимущества и недостатки существующих метрик для планирования ресурсов [17], была разработана методика комплексного измерения общего уровня дисбаланса системы, а также среднего уровня дисбаланса каждого сервера. Рассмотрены следующие критерии:

1. Средняя загрузка каждого u-го процессора $CPU_i^u(T)$ i-го сервера определяется как средняя загрузка процессора в течение наблюдаемого периода. Например, если период наблюдений составляет 1 мин., а загрузка процессора записывается через каждые 10 секунд, то есть CPU_i^u это среднее значение из шести записанных значений i-го сервера.

Аналогично определяется средняя загрузка каждой r-й памяти $RAM_i^r(T)$ i-го сервера и средняя загрузка k-го канала $Net_i^k(T)$ i-го сервера в течение наблюдаемого периода.

2. Поскольку измеренная загрузка процессора, использование памяти и канала потоком класса qs отличаются от средней загрузки процессора CPU_i^{qs}, памяти RAM_i^{qs} и канала Net_i^{qs} потоком класса qs отсутствием времени работы операционной системы и переключением между задачами, то можно ввести величины CPU_i^{qsv}, RAM_i^{qsv} и канала Net_i^{qsv} которые обозначают загрузку процессора, памяти и канала потоками класса qs, измеренную системой учета или монитором операционной системы.

Загрузка процессора потоком класса qs вычисляется по формуле:

$$CPU_i^{qs} = CPU_i^u \times f_{CPU}^{qs}, \quad (1)$$

где f_{CPU}^{qs} - доля суммарного использования u-го процессора, которую можно отнести на класс qs. Параметр f_{CPU}^{qs} вычисляется следующим образом: $f_{CPU}^{qs} = CPU_i^{qsv} \Big/ \sum_{\forall qs} CPU_i^{qsv}$.

Точно так же рассчитываются значения загрузки памяти и пропускной способности сети с учетом классов потоков. Загрузка памяти потоком класса qs вычисляется по формуле:

$$RAM_i^{qs} = RAM_i^r \times f_{RAM}^{qs}, \quad (2)$$

где f_{RAM}^{qs} - доля суммарного использования r-й памяти, которую можно отнести на класс qs. Параметр f_{RAM}^{qs} вычисляется следующим образом: $f_{RAM}^{qs} = RAM_i^{qsv} \Big/ \sum_{\forall qs} RAM_i^{qsv}$.

Загрузка канала потоком класса qs вычисляется по формуле:

$$Net_i^{qs} = Net_i^k \times f_{Net}^{qs} \quad , \tag{3}$$

где f_{Net}^{qs} - доля суммарного использования k-го канала, которую можно отнести на класс qs. Параметр f_{Net}^{qs} вычисляется следующим образом: $f_{Net}^{qs} = Net_i^{qsv} \Big/ \sum_{\forall qs} Net_i^{qsv}$.

3. Введем средний коэффициент использования всех процессоров в системе. Пусть $CPU_i^{n_i}$ средняя загрузка ЦПУ i-го сервера:

$$CPU_u^{All} = \frac{\sum_i^N CPU_i^u CPU_i^{n_i}}{\sum_i^N CPU_i^{n_i}}, \tag{4}$$

где N – общее число физических серверов в системе, n_i - количество ЦПУ на i-м сервере. Аналогичным образом средний коэффициент использования памяти $RAM_i^{m_i}$, пропускная способность линии связи $Net_i^{k_i}$ i-го сервера, вся память RAM_r^{All} и вся пропускная способность сети в системе Net_k^{All} может быть определена, как:

$$RAM_r^{All} = \frac{\sum_i^N RAM_i^r RAM_i^{m_i}}{\sum_i^N RAM_i^{m_i}}, \tag{5}$$

$$Net_k^{All} = \frac{\sum_i^N Net_i^k Net_i^{k_i}}{\sum_i^N Net_i^{k_i}}, \tag{6}$$

4. Значение дисбаланса всех процессоров. Используя формулу дисперсии, значение дисбаланса всех процессоров в системе определяется как:

$$ISL_{CPU} = \sum_i^N (CPU_i^u - CPU_u^{All})^2, \tag{7}$$

Точно так же могут быть рассчитаны значения дисбаланса памяти и пропускной способности сети:

$$ISL_{RAM} = \sum_i^N (RAM_i^r - RAM_r^{All})^2, \tag{8}$$

$$ISL_{Net} = \sum_i^N (Net_i^k - Net_k^{All})^2. \tag{9}$$

5. Введем комплексное значение дисбаланса нагрузки SIL_i i-го сервера, учитывающее все три ресурса сервера. Используя формулу расчета дисперсии как меры неравномерности, интегрированное значение дисбаланса нагрузки i-го сервера можем определить как:

$$SIL_i = a(CPU_i^u - CPU_u^{All})^2 + b(RAM_i^r - RAM_r^{All})^2 + c(Net_i^k - Net_k^{All})^2. \tag{10}$$

Параметры a, b, c обозначают весовые коэффициенты для процессора, памяти и пропускной способности сети, соответственно, выбираются экспериментальным

путем таким образом, что $a+b+c=1$ в зависимости от решаемых задач и структуры системы.

SIL_i применяется для обозначения уровня дисбаланса нагрузки путем сравнения коэффициентов использования процессора, памяти и пропускной способности сети.

Тогда суммарные значения дисбаланса всех серверов в системе записывается как:

$$ISL_{tot} = \frac{1}{N}\sum_i^N SIL_i \,. \qquad (11)$$

6. Средняя продолжительность работы при одинаковом количестве задач позволяет сравнивать различные алгоритмы планирования.

7. Период обработки на i-м сервере определяется как максимальная нагрузка на i-м сервере. Период обработки в системе определяется как средняя загрузка на всех серверах.

8. Эффективность использования определяется как средняя нагрузка на любом сервере.

Таким образом, для планирования ресурсов была разработана методика комплексного измерения общего уровня дисбаланса системы, а также среднего уровня дисбаланса каждого сервера.

Выводы

В работе было предложено решение актуальной научной задачи оценки загрузки узлов распределенной системы. Предложенный метод основан на расчете загрузки процессора, памяти и пропускной способности канала потоками разных классов обслуживания. Также введено комплексное значение дисбаланса нагрузки сервера, учитывающее весовые коэффициенты для процессора, памяти и пропускной способности сети. Соответственно, данный метод позволяет вычислить дисбаланс всех серверов системы, среднюю продолжительность работы при различных алгоритмах балансировки и эффективность использования ресурсов системы.

Список литературы:

1. *Cardellini V.* Dynamic Load Balancing on Web-server Systems / Valeria Cardellini, Michele Colajanni, Philip S. Yu // IEEE Internet Computing. - Vol.3. - No.3. – 1999. – P.28-39.

2. *Тарасов В.Н., Полежаев П.Н., Шухман А.Е., Ушаков Ю.А., Коннов А.Л.* Математические модели облачного вычислительного центра обработки данных с использованием Openflow // Вестник ОГУ. – №9 (145). – 2012. – С. 150-155.

3. *Игнатенко Е.И., Бессараб В.И., Дегтяренко И.В.* Адаптивный алгоритм мониторинга загруженности сети кластера в системе балансировки нагрузки. // Наукові праці ДонНТУ. – Вип. 21(183). – 2011. – С. 95-102.

4. *Chen H., Wang F., Helian N., Akanmu G.* User-priority guided min-min scheduling algorithm for load balancing in cloud computing // National Conference on Parallel Computing Technologies (PARCOMPTECH). – Bangalore, 2013. – P. 1-8.

5. *Cardellini V.* A performance study of distributed architectures for the quality of web services. // Proceedings of the 34th Conference on System Sciences. – Vol. 10. – 2001. – P.213-217.

6. *Keshav S.* An Engineering Approach to Computer Networking // Addison-Wesley, Reading, MA. – 1997. – P. 215-217.

7. *Liu J., Luo X., Zhang X., Zhang F., Li B.* Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm // IJCSI International Journal of Computer Science. – V.10(1). - № 3. - 2013. – P.134-139.

8. *Kameda H., Li L., Kim C., Zhang Y.* Optimal Load Balancing in Distributed Computer Systems. – London: Springer, Verlag London Limited. - 1997. – 238 p.

9. *Kaur R., Luthra P.* Load Balancing in Cloud Computing // Proc. of Int. Conf. on Recent Trends in Information, Telecommunication and Computing. – Association of Computer Electronics and Electrical Engineers. – 2014. – P. 374-381.

10. *Roth G.* Server load balancing architectures, Part 1: Transport-level load balancing. – 2008. – Режим доступа: http://www.javaworld.com/article/2077921/architecture-scalability/server-load-balancing-architectures--part-1--transport-level-load-balancing.html.

11. *Gupta R.O., Champaneria T.* A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment // International Journal of Advanced Research in Computer Science and Software Engineering. – 2013. – Vol.3(11).– P. 782-790.

12. *Kashyap D., Viradiya J.* A Survey Of Various Load Balancing Algorithms In Cloud Computing. // International Journal Of Scientific & Technology Research. – 2014. – Vol. 3(11). – P.115-119.

13. *Ghuge K., Doorwar M.* A Survey of Various Load Balancing Techniques and Enhanced Load Balancing Approach in Cloud Computing // International Journal of Emerging Technology and Advanced Engineering. – 2014. – Vol. 4(10). – P. 410-414.

14. *Anand C., Kapadia A.* Load Balancing Algorithm for Azure Virtualization with Specialized VM // International Journal of Innovations in Engineering and Technology (IJIET). – 2014. – Vol. 2(3). – P.216-223.

15. *Elzeki O., Reshad M., Elsoud M.*. Improved max-min algorithm in cloud computing // International Journal of Computer Applications. – 2012. – Vol. 50(12). – P. 22–27.

16. *Hu Y., Blake R., Emerson D.* An optimal migration algorithm for dynamic load balancing // Concurrency and Computation: Practice and Experience. – 1998. – Vol. 10 (6). – P. 467-483.

17. *Tian W., Zhao Y.* Optimized Cloud Resource Management and Scheduling: Theories and Practices. – Morgan Kaufman, 2014. – 284 p.