УДК 004.942

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ МЕТОДОВ ЛИНЕЙНОЙ РЕГРЕССИИ ДЛЯ ПОСТРОЕНИЯ АДАПТИВНОЙ СИСТЕМЫ УПРАВЛЕНИЯ В ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ СРЕДАХ ВИРТУАЛИЗАЦИИ

А. О. Ферубко, аспирант БГИТУ, Брянск, Россия;
orcid.org/0009-0006-5625-137X, e-mail: ferubko1999@yandex.ru
О. Д. Казаков, к.э.н, доцент, заведующий кафедрой ИТ БГИТУ, Брянск, Россия;
orcid.org/0000-0001-9665-8138, e-mail: kazakov@bgitu.ru

Рассматривается задача прогнозирования потребления оперативной памяти (ОП) в виртуализированной среде на базе KVM/QEMU для целей адаптивного управления ресурсами, в частности ОП, в гомогенной системе. ***Целью работы*** *является сравнительный анализ моделей линейной регрессии для построения прогноза потребления ОП как гостевой, так и хостовой операционных систем (ОС). Прогноз строится на основе обучающей выборки, которая формируется из данных о потреблении ОП, в предыдущие точки принятия системой решения о перераспределении ОП. Модель, принимающая решение опирается не на сами прогнозы потребления, а на объёмы свободной (не задействованной) ОП, значение которой получается вычитанием из всей имеющейся у подсистемы ОП той ОП, которая по прогнозу будет задействована в процессах подсистемы. И на основе прогноза незадействованной ОП формируется обоснованное решение о необходимости и масштабе перераспределения ОП между подсистемами. Решение зависит не только от прогнозов, но и от величины окна, предохраняющего систему от слишком частых перераспределений в условиях относительно небольшой разницы, а также от самих ОС и их ограничений на минимальный объём ОП.*

Ключевые слова: *оперативная память, KVM, QEMU, гипервизор, виртуализация, гипервизор первого типа, линейная регрессия, регуляризация, регуляризация по Тихонову, L1-регуляризация, взвешенная линейная регрессия, временные ряды, операционные системы, гомогенные системы, машинное обучение.*

DOI: 10.21667/1995-4565-2026-95-73-84

Введение

Важное место в современной ИТ инфраструктуре занимают гипервизоры [1, 2], которые применяют благодаря их высокой отказоустойчивости, а также, степени использования и изоляции ресурсов [3]. Например, по прогнозам [4] рынок гипервизоров к 2035 году составит 13.1 млрд долларов. Они применяются при построении облачных провайдеров, виртуальных машин (ВМ), систем удалённых рабочих столов (VDI), ЦОДов, а также в военной сфере [5]. Гипервизоры бывают разными, и от возможностей конкретного гипервизора напрямую зависят производительность, отказоустойчивость и общая эффективность системы в целом. По устройству гипервизоры классифицируют на две группы – гипервизоры первого и второго типа. Первые умеют напрямую взаимодействовать с аппаратным обеспечением без использования основной (хостовой) ОС, что позволяет добиться высокой производительности, отказоустойчивости и изоляции. Гипервизорам второго типа нужна хостовая ОС для взаимодействия с аппаратным обеспечением, из-за чего они уступают в производительности гипервизорам первого типа по причине наличия дополнительного слоя абстракций в виде хостовой ОС [6]. С целью добиться большей производительности в данной статье рассматривались только гипервизоры первого типа. Гипервизоры также классифицируются по доступности исходного кода на гипервизоры с открытым исходным кодом и на проприетарные. Данный фактор влияет на возможность полного понимания процессов, идущих внутри ВМ, поэтому рассматривались только гипервизоры с открытым исходным кодом. KVM [7] является

одним из самых популярных гипервизоров первого типа с открытым исходным кодом. К его достоинствам относят высокую производительность и низкую задержку при работе с гостевыми ОС [8]. Также к его плюсам можно отнести то, что он использует ядро Linux, что даёт возможность использовать имеющиеся драйверы для аппаратного обеспечения, а также гарантирует хорошую работу в качестве гостевых ОС, российских ОС, таких как Alt Linux, РЕД ОС, Astra Linux, и прочих ОС основанных на Linux. Также он обладает минимальной сложностью разворачивания, что качественно выделяет его на фоне других конкурентов. Например, гипервизор Xen [7] тоже обладает хорошими эксплуатационными характеристиками, но на прядок сложнее в деплое [9]. Например, там можно выбрать 4 механизма виртуализации [10]. Другой набирающий популярность гипервизор – Rust-Shyper [11] пока имеет недостаточно материалов и инструментов для полноценного применения. Поэтому в данной статье рассматривается гипервизор KVM. Для эмуляции ресурсов вместе с KVM часто используют QEMU [12]. Но у гипервизоров присутствует дефицит адаптивных систем управления ресурсами, в частности ОП, которые бы могли в режиме реального времени анализировать потребление ресурсов и перераспределять их при необходимости. Данная проблема продемонстрирована на примере системы, где основная ОС – Ubuntu, а гостевая ОС – Fedora (рисунок 1).

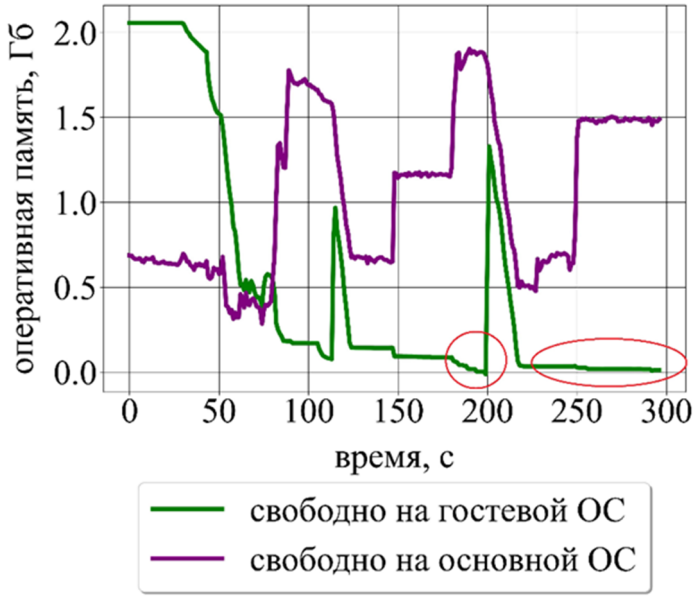

Рисунок 1 – Пример распределения свободной ОП в системе, где основная ОС – Ubuntu, а гостевая ОС – Fedora. Красными окружностями обозначены области дефицита ОП у одной из подсистем при условии её дефицита в другой подсистеме
Figure 1 – An example of free software distribution in a system where the main OS is Ubuntu and guest OS is Fedora. Red circles indicate OP deficiency areas in one of subsystems, subject to its deficiency in other subsystem

Из анализа рисунка 1 можно сделать вывод, что без применения адаптивных систем управления возможны сценарии нерационального распределения ОП между различными подсистемами системы. Одним из методов построения адаптивной системы управления является модель, использующая аппарат регрессионного анализа для предсказания потребления ОП и на основе полученных цифр производить расчёт объёма свободной ОП в каждой подсистеме и на основании анализа полученных цифр, а также ограничений подсистем (минимальный объём ОП, необходимый для функционирования ОС) производить перераспределение ОП между подсистемами в режиме реального времени без перезагрузки (гипервизор KVM умеет изменять объём ОП без перезагрузки гостевой ОС). Но существуют разные виды регрессий и разные сценарии нагрузки, что создает задачу выбора подходящей модели ре-

грессии. В статье рассматриваются только модели линейной регрессии, так как они обладают относительно невысокой вычислительной сложностью.

Цель работы – сравнительный анализ моделей линейной регрессии для построения прогноза потребления ОП как гостевой, так и хостовой операционных систем (ОС) для построения адаптивной системы управления гипервизором.

Постановка задачи

Необходимо исследовать и сравнить различные методы линейной регрессии для предсказания потребления ОП. В качестве объекта исследования выбрана система, в которой в роли основной ОС выступает ОС Ubuntu 24.04, а в качестве гостевой ОС выступает Fedora Linux 40, в качестве гипервизора выступает KVM с QEMU для эмуляции аппаратного обеспечения. Для сбора данных о потреблении ОП и изменения объёма ОП в гостевой ОС будет использован инструмент Libvirt [13], а в основной ОС psutil [14]. Задача состоит в том, чтобы на основе оценки качества предсказания потребления ОП при различных режимах нагрузки различными линейными моделями выбрать модель, подходящую для построения адаптивной системы управления гипервизором. В ходе исследования будут рассмотрены алгоритмы линейной регрессии, L1-регуляризованной линейной регрессии, L2-регуляризованной линейной регрессии (регуляризация по Тихонову), взвешенная линейная регрессия с метрическим ядром, взвешенная линейная регрессия с экспоненциальным ядром. Дополнительно будет исследован процесс оптимизации гиперпараметров моделей, так как правильная настройка гиперпараметров может существенно повысить точность предсказаний [15] и качество самой модели.

Теоретическая часть

Линейным моделям для предсказания необходима обучающая выборка, поэтому в процессе функционирования системы (рисунок 2) необходимо разделить на два этапа – накопление стартового набора обучающих данных (этап 1, рисунок 2) и эксплуатация (этапы 2 – 6, рисунок 2). Во время первого этапа модель управления накапливает данные о системе (сколько ОП используется/свободно в каждой из подсистем). В данном исследовании для обучения линейных моделей использовался массив данных потребления ОП, который собирался в течение пяти секунд (с секундным интервалом). Соответственно первый этап длился пять секунд. Данные потребления ОП имеют структуру временного ряда. Роль независимой переменной (предикторов) выполняли центрированные данные времени в секундах, роль зависимой переменной (откликов) выполняли данные потребления ОП. Во время этапа эксплуатации перед каждым шагом принятия решения моделью о необходимости и масштабе перераспределения ОП происходили актуализация обучающей выборки – в неё добавлялись более свежие значения, а более старые удалялись таким образом, чтобы длина обучающей выборки равнялась пяти. В работе использовались различные варианты линейной регрессии.

Линейная регрессия – это базовый алгоритм регрессионного анализа, который моделирует линейную зависимость между предикторами и откликами. Модель выражается уравнением $y = ax + b + \varepsilon$, где x – это предиктор, y – отклик, а ε – величина случайной ошибки. Суть метода заключается в том, чтобы найти такие оценки a и b, при которых сумма квадратов ошибок будет минимальной [16].

Реальные данные редко являются гладкими монотонными функциями – им свойственны различные неточности и погрешности. В ОС это происходит из-за большого числа различных процессов, выполняемых параллельно. Алгоритмы машинного обучения, стремясь максимально точно научиться предсказывать обучающие данные, невольно могут начать ошибаться из-за неидеальности данных. Ибо они, опираясь лишь на данные потребления ОП, не могут верно воспроизвести сложную структуру процессов, происходящих внутри подсистемы. Регуляризация – это один из приёмов, помогающих избежать переобучения модели. За счёт ограничения сложности модели, при помощи добавления штрафа за сложность [17]. Од-

ним из подходов к регуляризации является L1-регуляризация, которая заключается в минимизации квадрата ошибок при условии, что сумма абсолютных значений коэффициентов меньше заданной константы. То есть модель накладывает штраф на величину абсолютных значений коэффициентов. Поэтому чем меньше значение константы, тем выше штраф [18]. Другой подход к регуляризации основан на идее минимизации суммы квадратов ошибок и некоего слагаемого, получаемого произведением константы на норму от предикторов. Данный подход называется L2-регуляризованной (регуляризация по Тихонову [19]) регрессией. Регуляризация в этом подходе пропорциональна значению константы. В случае, если константа штрафа равна нулю, то задача сводится к простому методу наименьших квадратов [20]. К достоинствам данных методов можно отнести то, что они не так сильно восприимчивы к неидеальности данных, которая может возникать из-за большого числа процессов с малой длительностью жизненного цикла и малым потреблением ОП, что позволит модели не терять обобщающую способность из-за наличия в системе множества параллельных процессов.

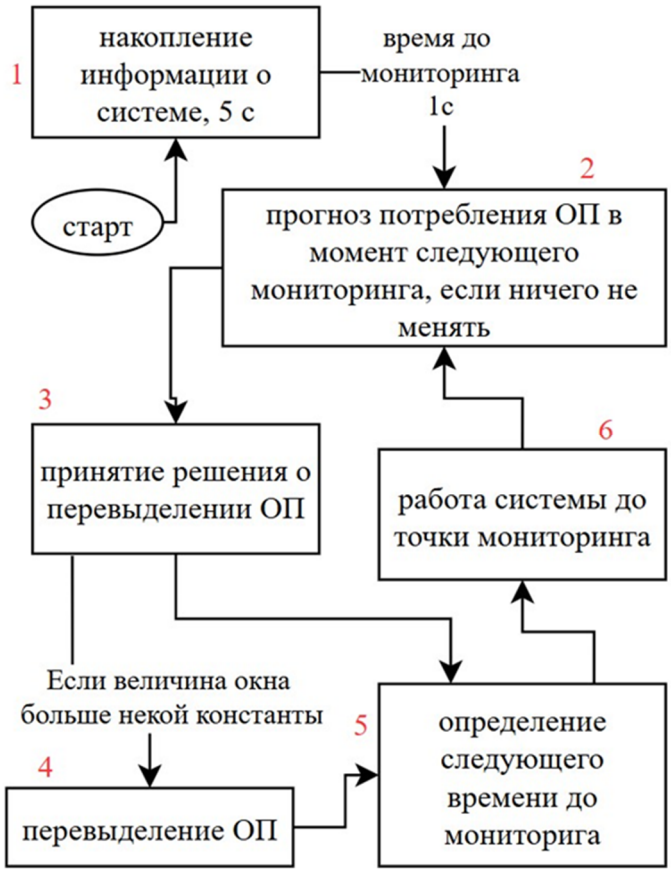

Рисунок 2 – Схема работы адаптивной системы управления гипервизором. Красными цифрами на схеме обозначены этапы функционирования модели
Figure 2 – The scheme of adaptive hypervisor management system. Red numbers on the diagram indicate model operation stages

Метод взвешенной линейной регрессии связан с таким понятием, как гетероскедастичность [21], когда вклад различных наблюдений в итоговую модель неодинаков. Существуют разные подходы к определению влияния конкретных предикторов на итоговую модель. Один из подходов заключается в использовании весовых коэффициентов (ВК), нормированных на сумму ВК обучающего набора. Все ВК обратно пропорциональны расстоянию от точки, в которой нужно предсказать значение зависимой переменной, до точки из обучающего набора, для которой рассчитывают ВК, в степени n [22]. Также существует версия данного алгоритма, в которой расстояние является частью выражения показателя степени, в которую возводят число Эйлера. В выражение показателя помимо расстояния в степени n входит некий

параметр, который иногда называют окном [23, 24]. Оба алгоритма основаны на идее того, что вклад точки в итоговую модель обратно пропорционален расстоянию этой точки до точки, в которой будет производиться предсказание отклика. Данная особенность позволяет системе быстрее реагировать на старт ресурсоёмких процессов. Но, это может быть опасно тем, что модель может слишком сильно реагировать на кратковременный скачок потребления ОП, предсказав некорректное потребление ОП. Поэтому данный алгоритм является довольно восприимчивым к не идеальности данных, но именно он может быть полезен при резкой смене режима работы. Например, когда ОС относительно длительное время потребляла постоянное значение ОП и резко в ней запустили ресурсоёмкий процесс. В такой ситуации алгоритмы взвешенной регрессии имеют ряд преимуществ над алгоритмами регуляризованной регрессии, которые в подобной ситуации отреагируют даже медленнее чем простая линейная регрессия. Но в ситуации, когда в гипервизоре постоянно запускаются и прерываются огромное число различных процессов с различным потреблением ОП и профиль потребления ОП имеет большое количество экстремумов на небольшом временном интервале, именно регуляризованная регрессия обладает преимуществом.

Для оценки качества моделей и оптимизации гиперпараметров использовали следующие метрики:

– квадратный корень из среднеквадратичной ошибки (RMSE) [25]

– коэффициент детерминации (Q^2)[15].

Экспериментальная часть

Была создана система на языке программирования Python 3.12 с применением библиотек libvirt, psutils, sklearn, numpy. Для эмуляции аппаратного обеспечения применяли QEMU. В качестве основной ОС выступала Ubuntu 24.04, а в качестве гостевой ОС выступала Fedora Linux 40, потому что обе эти системы поддерживают динамическую работу с ОП и могут менять её объём без перезагрузки. Также при заполнении xml файла конфигурации понадобилось указать параметр hotplug и указать модель ОП – DIMM [26,27]. Система работает по схеме, изображённой на рисунке 2. Её функционирование начинается со старта и накопления информации о системе (этап 1). Этот этап нужен для того, чтобы у модели были данные для обучения регрессии. Одним из параметров системы является величина времени между точками мониторинга, в которых и принимается решение о перераспределении ОП. На старте считаем эту величину равной 1 секунде. Также среди системных переменных присутствует размер окна, влияющего на итоговое решение модели о перераспределении ОП. Размер этого окна равен 10 МБ. Было выбрано данное значение, так как при нём довольно удобно менять режим нагрузки на систему. Если взять сильно маленькое значение, то даже минимальный скачок потребления будет восприниматься моделью как старт нового ресурсоёмкого процесса, что не позволит проверить систему при стабильном режиме нагрузки (когда потребление ОП практически не меняется). Если же взять слишком большое значение, то будет трудно проверить поведение системы при нестабильном режиме нагрузки (когда в системе стартуют и прерываются различные ресурсоёмкие процессы).

У системы бывают разные режимы нагрузки – когда, например, в системе запущен один процесс с постоянным потреблением ресурсов, а остальные процессы на его фоне пренебрежимо малы, то тогда можно считать, что система работает в стабильном режиме, а значит, можно увеличить интервал времени между точками мониторинга. А если в системе постоянно стартуют и прерываются различные тяжеловесные процессы и профиль нагрузки имеет сложный рельефный вид, то, тогда наоборот – важно чаще производить операцию мониторинга и максимально оперативно перераспределять ресурсы в пользу более нуждающейся в них подсистемы. Поэтому в модели адаптивного управления присутствует этап определения следующего времени до мониторинга (этап 5 на рисунке 2). Среди переменных системы управления присутствует массив, в котором хранится история принятия решений моделью о перераспределение ОП за предыдущие 5 циклов. Благодаря анализу этого массива можно

выяснить режим нагрузки на систему. Если все пять раз модель решала, что необходимо перераспределение ОП, то система классифицируется как нестабильная и устанавливается минимальное время до следующего мониторинга (в рамках данной работы это 1 с). Если же модель менее пяти раз принимала решение о перераспределении ОП, то, значит, режим нагрузки на систему чуть более постоянный. Зависимость времени до следующего мониторинга от количества принятых решений о перераспределении ОП, использованная при написании данной статьи, приведена в таблице 1. Решение о перераспределении зависит ещё и от величины окна. Если разница по модулю между прогнозами свободной ОП всех подсистем меньше значения окна, то тогда система принимает решение не перевыделять ОП. А если разница хотя бы для любых двух подсистем больше значения окна и при этом ни одна из них не упирается в ограничения гостевых ОС по минимальному объёму ОП, то тогда система принимает решение о перераспределении ОП. Алгоритм перераспределения основан на выделении такого количества ОП, чтобы в каждой из подсистем было одинаковое количество свободной ОП.

Таблица 1 – Зависимость времени до следующего мониторинга от количества перераспределений ОП в обучающем наборе данных

Table 1 – Time dependence till next monitoring from the number of RAM redistributions in training dataset

Количество перераспределений	Время до следующего мониторинга, с
0	10
1	8
2	6
3	4
4	2
5	1

На этапе 2 (см. рисунок 2), шёл подбор различных регрессионных моделей. Были проведены эксперименты для 5 моделей линейной регрессии. Результаты анализа приведены в таблице 2. Там, где требовалось, производили оптимизацию гиперпараметров. Приводили пример графика с оптимизацией гиперпараметров взвешенной линейной регрессии с метрическим ядром на рисунке 3. Результаты поместили в таблицу 2.

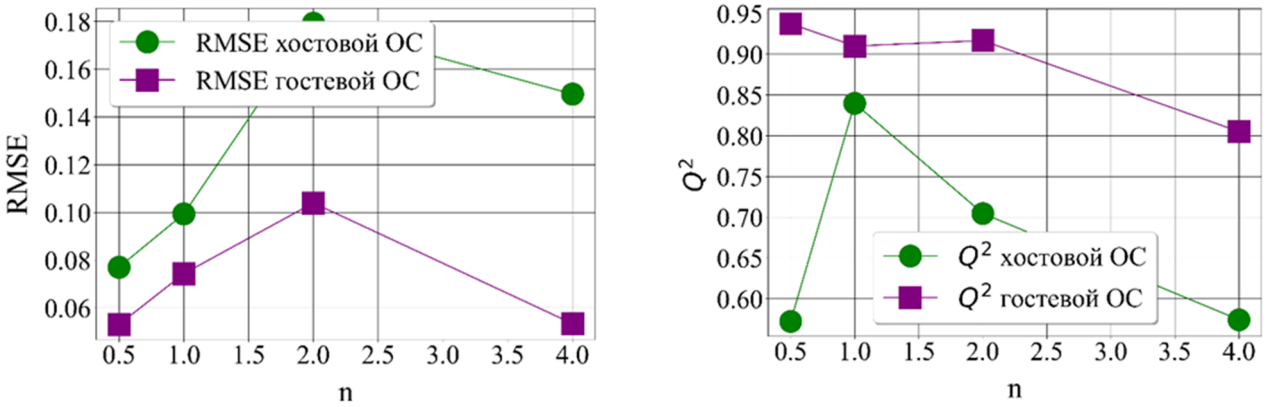

Рисунок 3 – Оптимизация гиперпараметров для взвешенной линейной регрессии с метрическим ядром. Слева RMSE, справа коэффициент детерминации

Figure 3 – Hyperparameter optimization for weighted linear regression with metric kernel. RMSE on the left, determination coefficient on the right

Простая линейная регрессия показала себя неплохо. Она быстро реагировала на изменения режима нагрузки. Модель линейной регрессии была реализована с применением библиотеки sklearn на языке программирования Python 3.12. Модель справилась с поставленной перед ней задачей. Были ликвидированы ситуации, когда одна из подсистем испытывает дефицит ОП, в то время как другая подсистема имеет её в избытке.

Таблица 2 – Параметры различных регрессионных моделей, применявшихся для построения адаптивной системы управления гипервизором KVM

Table 2 – Parameters of various regression models used to build adaptive KVM hypervisor management system

Модель	Гипер-параметр	Значение	RMSE гостевой ОС, МБ	Q^2 гостевой ОС	RMSE хостовой ОС, МБ	Q^2 хостовой ОС
Линейная регрессия	-	-	101	0,93	70	0,70
L1-регуляризованная регрессия	α	10	65	0,95	62	0,74
L2-регуляризованная регрессия	β	0.1	58	0,96	60	0,82
Взвешенная линейная регрессия с метрическим ядром	n	1	53	0,93	70	0,84
Взвешенная линейная регрессия с экспоненциальным ядром	h	0.5	55	0,90	173	0,55

Модель L1-регуляризованной регрессии была реализована при помощи библиотеки sklearn. Для неё производилась оптимизация гиперпараметров по логарифмической шкале. Результаты применения L1-регуляризованной регрессия являются неоднозначными. Модель действительно реже реагировала на кратковременные процессы с низким потреблением ОП, но модель также плохо реагировала и на старт ресурсоёмких процессов после длительного режима постоянной нагрузки. Из-за этого пришлось ограничить сверху область значений параметра регуляризации. Негативный эффект усиливается, если в подсистеме присутствует мало свободной ОП. Некоторые ресурсоёмкие процессы, например запуск веб-браузера Firefox, просто не запускались, так как в процессе их запуска не происходило увеличение ресурсов, адаптивная модель не откликалась на резкий рост их потребления и не производила перераспределение.

Похожим образом обстоят дела и с L2-регуляризованной моделью. Для модели с L2-регуляризацией Тихонова также производили оптимизацию гиперпараметра по логарифмической шкале. Применение обоих регуляризованных моделей возможно лишь при низкой регуляризации, когда регуляризация оказывает слабое влияние на работу линейной модели. Но, L2-регуляризованная модель показала себя лучше, чем L1-регуляризованная регрессия.

Чуть иначе обстоят дела с моделями взвешенной линейной регрессии. И модель с метрическим ядром, при помощи которого веса рассчитываются по формуле

$$w_i = \frac{\left(t - t_i\right)^n}{\sum_{j=1}^{5}\left(t - t_j\right)^n},$$

где t – момент времени следующего мониторинга, t_j – момент времени одного из предыдущих мониторингов, n – гиперпараметр. И модель с экспоненциальным ядром, при помощи которого веса рассчитываются по формуле

$$w_i = \frac{\exp\left(-\frac{(t-t_i)^2}{2h^2}\right)}{\sum_{j=1}^{5}\exp\left(-\frac{(t-t_j)^2}{2h^2}\right)}?$$

где t – момент времени следующего мониторинга, t_j – момент времени одного из предыдущих мониторингов, h – гиперпараметр, хорошо реагировали на резкое изменение режима нагрузки, и в целом, хорошо справились со своей задачей, что продемонстрировано на примере рисунка 4. Но обе модели реагировали на старт короткоживущих не ресурсоёмких процессов, что немного ухудшает их результаты, причём вариант с экспоненциальным ядром чаще ошибался. Применение гетероскедастичности позволило уменьшить вклад в итоговую модель процессов в подсистеме, которые были давно и уже не вносят вклад в актуальное потребление, что хорошо. Но их результаты сложно назвать идеальными.

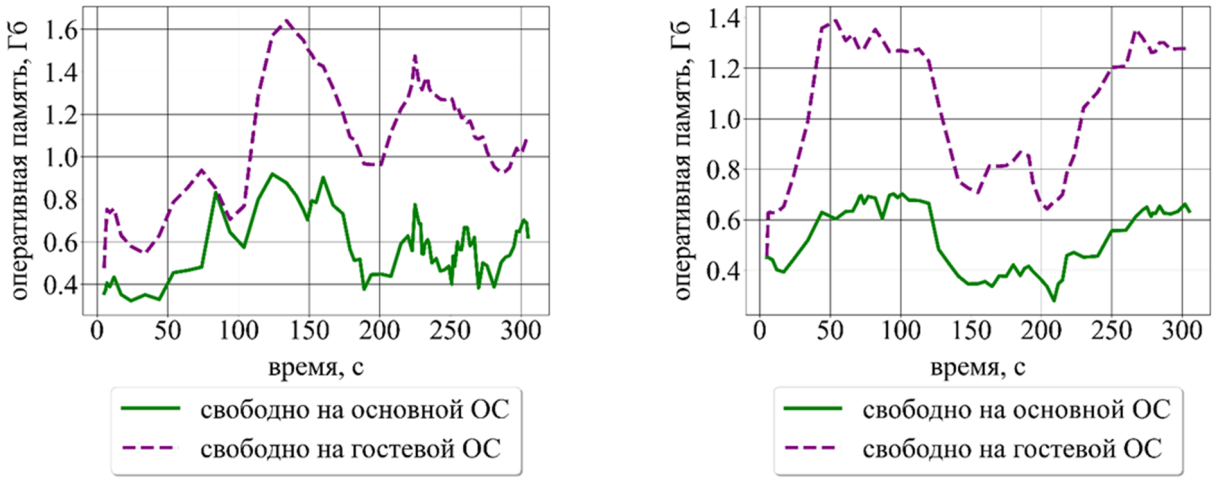

Рисунок 4 – Пример профиля свободной ОП для простой линейной регрессии (слева) и линейной регрессии с L2-регуляризацией по Тихонову (справа)
Figure 4 – Example of free OP profile for simple linear regression (left) and linear regression with Tikhonov L2 regularization (right)

Для всех пяти моделей замечено незначительное отставание по фазе от реального потребления ресурсов, что продемонстрировано на примере модели взвешенной линейной регрессии с метрическим ядром на рисунке 5. Причину данного отставания выяснить не удалось.

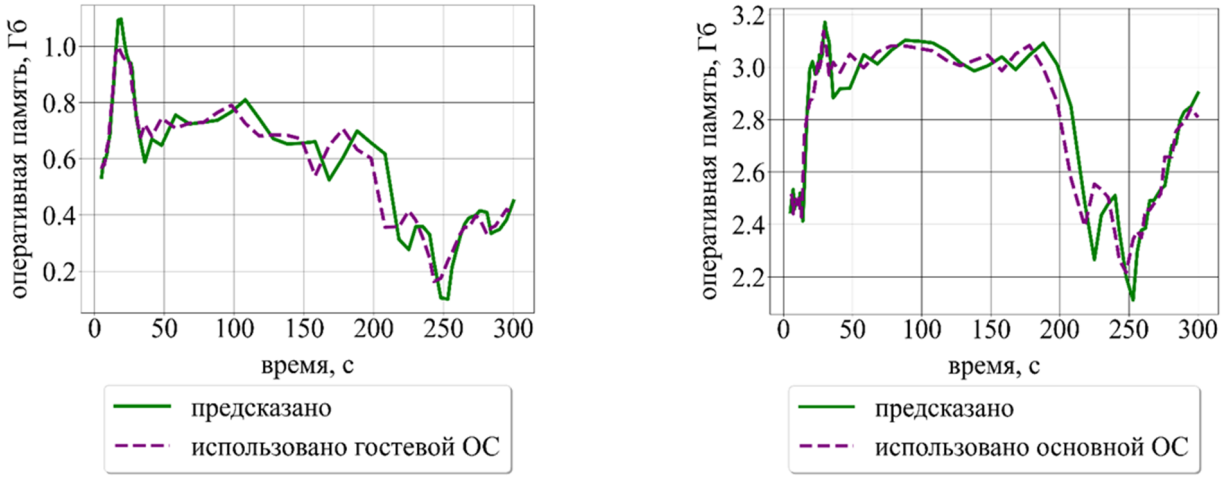

Рисунок 5 – Пример прогноза потребления ОП: слева для гостевой ОС, справа для основной ОС
Figure 5 – An example of RAM consumption forecast – on the left for guest OS, on the right for host OS

Опираясь на данные для гостевой и хостовой ОС и на комбинацию двух метрик, выяснили, что взвешенная линейная регрессия с метрическим ядром и L2-регуляризованная линейная регрессия являются более предпочтительными решениями для построения адаптивной системы управления.

Для гостевой ОС для всех моделей было получено значение Q^2 больше 0,9, что соответствует хорошим моделям, а по RMSE она незначительно опережает модель взвешенной линейной регрессии с экспоненциальным ядром и модель регуляризованную по Тихонову, что скорее говорит о превосходстве над моделью обычной линейной регрессии, чем над другими моделями.

Для хостовой ОС для всех моделей кроме взвешенной линейной регрессии с экспоненциальным ядром были получены практически идентичные значения RMSE, а вот для метрики Q^2 для взвешенной линейной регрессии с метрическим ядром и для L2-регуляризованной регрессии были получены результаты, свидетельствующие о высокой предсказательной способности обоих моделей.

Заключение

В ходе проведённого исследования была реализована система адаптивного управления гипервизором KVM, использующая различные алгоритмы линейной регрессии для предсказания потребления ОП, на основе чего, принималось обоснованное решение о перераспределении ресурсов или его отсутствии. Удалось минимизировать количество исходов, при которых одна из подсистем испытывает дефицит ресурсов, в то время как другие подсистемы испытывают его избыток. Удалось в модель внедрить классификатор режима работы системы, который позволил оптимизировать частоту мониторинга. Наиболее подходящей для построения системы адаптивного управления гипервизором были выбраны модели взвешенной линейной регрессии с метрическим ядром степени 1 и модель с L2-регуляризацией по Тихонову с параметром регуляризации, равным 0.1.

Но, при этом выявлена необходимость улучшения модели взвешенной линейной регрессии, чтобы уменьшить количество ложных срабатываний – когда модель принимает короткоживущий не ресурсоёмкий процесс за старт ресурсоёмкого и неверно рассчитывает потребление ОП, что приводит к ошибкам адаптивной модели управления при принятии решения о перераспределении ресурсов.

Была выявлена необходимость защиты регуляризованных моделей от ошибок детектирования старта роста потребления ОП после длительного периода равномерной нагрузки, что приводит к тому, что процессы не получают своевременно необходимые объёмы ресурсов и иногда даже могут завершаться из-за их нехватки.

Было выявлено отставание всех использовавшихся регрессионных моделей по фазе от реального потребления ОП. Причины этого отставания и методы его уменьшения будут рассмотрены в дальнейших исследованиях.

Результаты данного исследования имеют значительный потенциал для разработки новых решений в области адаптивных систем управления различными гипервизорами. Применение взвешенных линейных моделей и моделей с регуляризацией может привести к значительному повышению эффективности использования облачных ресурсов и уменьшить простои ресурсов и их неэффективное использование, что приведёт к дополнительному росту рентабельности применения гипервизоров, и повлечёт за собой рост их доли на ИТ рынке.

Библиографический список

1. **Dittner R., Rule D.** The Best Damn Server Virtualization Book Period: Including Vmware, Xen, and Microsoft Virtual Server. 1-st ed. Syngress, 2007. 960 P.

2. **De Alfonso C., Calatrava A., Moltó G.** Container-based virtual elastic clusters // Journal of Systems and Software. 2017. Vol. 127. Pp. 1-11.

3. **Asyabi E., Sharifi M., Bestavros A.** ppXen: A hypervisor CPU scheduler for mitigating performance variability in virtualized clouds // Future Generation Computer Systems. 2018. Vol. 83. Pp. 75-84.

4. https://www.marketresearchfuture.com/reports/desktop-hypervisor-market-4246

5. **Лимонцев Д.С.** Анализ российского рынка VDI-решений // Шарыгинские чтения: Международная научная конференция ведущих научных школ в области радиолокации, радионавигации и радиоэлектронных систем передачи информации. 2024. Т. 1, № 1. С. 417–423.

6. **Zhao N. et al.** Scheduling virtual machines and containers: A comparative review of techniques, performance, and future trends // Journal of Systems Architecture. 2025. Vol. 168. Pp. 103583.

7. **Abeni L., Faggioli D.** Using Xen and KVM as real-time hypervisors // Journal of Systems Architecture. 2020. Vol. 106. Pp. 101709.

8. **Cinque M. et al.** Virtualizing mixed-criticality systems: A survey on industrial trends and issues // Future Generation Computer Systems. 2022. Vol. 129. Pp. 315–330.

9. **Jones M. T.** Anatomy of a Linux hypervisor / пер. Ромоданов Н. 2009.

10. **Cinque M., De Simone L., Ottaviano D.** Temporal isolation assessment in virtualized safety-critical mixed-criticality systems: A case study on Xen hypervisor // Journal of Systems and Software. 2024. Vol. 216. Pp. 112-147.

11. **Mo C. et al.** Rust-Shyper: A reliable embedded hypervisor supporting VM migration and hypervisor live-update // Journal of Systems Architecture. 2023. Vol. 142. Pp. 102948.

12. **Marinescu D.C.** Cloud Computing Theory and Practice. Third Edition. USA: Elsevier, 2022. 653 p.

13. **Ashley W.D.** Foundations of Libvirt Development: How to Set Up and Maintain a Virtual Machine Environment with Python. Berkeley, CA: Apress. 2019. 408 p.

14. **Parvathy M., Antony Balasingam J., Sanjith E. S.** Real-Time Web Server Monitoring System using Python // Journal of Artificial Intelligence and Capsule Networks. 2024. Vol. 6, No. 3. Pp. 332-339.

15. **Корячко В.П., Викулин С.Д., Волков А.В.** Сравнительное исследование методов машинного обучения и нейронных сетей для предсказания химического состава материалов // Вестник Рязанского государственного радиотехнического университета. 2025. № 91. С. 50-63.

16. **Дрейпер Н., Смит Г.** Математико-статистические методы за рубежом. Второе издание / пер. Адлер Ю.П., Горский В.Г. М.: Финансы и статистика, 1986. Т. 1.

17. **Severson K. et al.** Elastic net with Monte Carlo sampling for data-based modeling in biopharmaceutical manufacturing facilities // Computers & Chemical Engineering. 2015. Vol. 80. Pp. 30-36.

18. **Nasehi Tehrani J. et al.** L1 regularization method in electrical impedance tomography by using the L1-curve (Pareto frontier curve) // Applied Mathematical Modelling. 2012. Vol. 36, No. 3. Pp. 1095-1105.

19. **Tikhonov A.I.** Solution of incorrectly formulated problems and the regularization method // Soviet Mathematics Doklady. 1963. No. 5. Pp. 1035-1038.

20. **Kim S.-J. et al.** An Interior-Point Method for Large-Scale-Regularized Least Squares // IEEE J. Sel. Top. Signal Process. 2007. Vol. 1, No. 4. Pp. 606-617.

21. **Hwang E., Hong W.-T.** A multivariate HAR-RV model with heteroscedastic errors and its WLS estimation // Economics Letters. 2021. Vol. 203. Pp. 109855.

22. **Tellinghuisen J.** Chapter 10 Least Squares in Calibration: Weights, Nonlinearity, and Other Nuisances // Methods in Enzymology. Academic Press, 2009. Vol. 454. Pp. 259-285.

23. **Chen C. et al.** Fast iteratively reweighted least squares algorithms for analysis-based sparse reconstruction // Medical Image Analysis. 2018. Vol. 49. Pp. 141-152.

24. **Miller S., Startz R.** Feasible generalized least squares using support vector regression // Economics Letters. 2019. Vol. 175. Pp. 28–31.

25. **Liemohn M.W. et al.** RMSE is not enough: Guidelines to robust data-model comparisons for magnetospheric physics // Journal of Atmospheric and Solar-Terrestrial Physics. 2021. Vol. 218. Pp. 105624.

26. **Mueller S.** Модернизация и ремонт ПК. 19-е изд. / пер. Тараброва И.Б. СПб: Издательский дом «Вильямс». 2011. 1072 с.

27. **Bruce J., Spencer W.N., David T.W.** Memory Systems Cache, DRAM, Disk. USA: Elsevier, 2008. 982 P.

UDC 004.942

COMPARATIVE STUDY OF LINEAR REGRESSION METHODS TO BUILD ADAPTIVE MANAGEMENT SYSTEM IN HIGH-PERFORMANCE VIRTUALIZATION ENVIRONMENTS

A. O. Ferubko, post-graduate student, BSTU, Bryansk, Russia;
orcid.org/0009-0006-5625-137X, e-mail: ferubko1999@yandex.ru
O. D. Kazakov, PhD, Associate Professor, Head of the Department, BSTU, Bryansk, Russia;
orcid.org/0000-0001-9665-8138, e-mail: kazakov@bgitu.ru

The problem of predicting RAM consumption in a virtualized environment based on KVM/QEMU for the purposes of adaptive resource management, RAM in particular, in a homogeneous system is considered. The aim is to find the main performance indicators of linear regression models for predicting RAM consumption of both guest and host operating systems (OS). The forecast is based on a training sample, which is formed from data on RAM consumption at previous points when the system made a decision on RAM redistribution. The decision-making model is based not on the consumption forecasts themselves, but on the volumes of free (unused) RAM, the value of which is obtained by subtracting from the entire RAM subsystem the RAM that is predicted to be involved in subsystem processes. And based on the forecast of unused RAM, an informed decision about the need and scale of RAM redistribution between subsystems is made. The decision depends not only on forecasts, but also on the size of the window that protects the system from too frequent redistributions in conditions of relatively small difference, as well as on the OS themselves and their restrictions on minimum amount of RAM.

Keywords: *RAM, KVM, QEMU, hypervisor, virtualization, hypervisor of the first type, linear regression, regularization, Tikhonov L2-regularization, L1-regularization, weighted linear regression, time series, operating systems, homogeneous systems, machine learning.*

DOI: 10.21667/1995-4565-2026-95-73-84

References

1. **Dittner R., Rule D.** *The Best Damn Server Virtualization Book Period: Including Vmware, Xen, and Microsoft Virtual Server*. 1-st ed. Syngress, 2007. 960 p.

2.**De Alfonso C., Calatrava A., Moltó G.** Container-based virtual elastic clusters. *Journal of Systems and Software*. 2017, vol. 127, pp. 1-11.

3. **Asyabi E., Sharifi M., Bestavros A.** ppXen: A hypervisor CPU scheduler for mitigating performance variability in virtualized clouds. *Future Generation Computer Systems*. 2018, vol. 83, pp. 75-84.

4. https://www.marketresearchfuture.com/reports/desktop-hypervisor-market-4246

5. **Limonze D.S.** Analiz rossijskogo rynka VDI-reshenij. *Sharyginskie chteniya: Mezhdunarodnaya nauchnaya konferenciya vedushchih nauchnyh shkol v oblasti radiolokacii, radionavigacii i radioelektronnyh sistem peredachi informacii*. 2024, vol. 1, no. 1. Pp. 417–423. (in Russian).

6. **Zhao N. et al.** Scheduling virtual machines and containers: A comparative review of techniques, performance, and future trends. *Journal of Systems Architecture*. 2025, vol. 168, pp. 103583.

7. **Abeni L., Faggioli D.** Using Xen and KVM as real-time hypervisors. *Journal of Systems Architecture*. 2020, vol. 106, pp. 101709.

8. **Cinque M. et al.** Virtualizing mixed-criticality systems: A survey on industrial trends and issues. *Future Generation Computer Systems*. 2022, vol. 129, pp. 315-330.

9. **Jones M.T.** *Anatomy of a Linux hypervisor* / per. Romodanov N. 2009.

10. **Cinque M., De Simone L., Ottaviano D.** Temporal isolation assessment in virtualized safety-critical mixed-criticality systems: A case study on Xen hypervisor. *Journal of Systems and Software*. 2024, vol. 216, pp. 112147.

11. **Mo C. et al.** Rust-Shyper: A reliable embedded hypervisor supporting VM migration and hypervisor live-update. *Journal of Systems Architecture*. 2023, vol. 142, pp. 102948.

12. **Marinescu D.C.** *Cloud Computing Theory and Practice. Third Edition*. USA: Elsevier, 2022. 653 p.

13. **Ashley W.D.** *Foundations of Libvirt Development: How to Set Up and Maintain a Virtual Machine Environment with Python. Berkeley.* CA: Apress, 2019. 408 p.

14. **Parvathy M., Antony Balasingam J., Sanjith E.S.** Real-Time Web Server Monitoring System using Python. *Journal of Artificial Intelligence and Capsule Networks*. 2024, vol. 6, no. 3, pp. 332-339.

15. **Koryachko V.P., Vikulin S.D., Volkov A.V.** Comparative study of machine learning methods and neural networks for predicting chemical composition of materials. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta*. 2025, no. 91, pp. 50-63. (in Russian).

16. **Draper N.R., Smith H.** *Applied Regression Analysis*. 2-nd ed. / translate. Adler Y.P., Gorskij V.G. Moscow: Finansy i statistika. 1986, vol. 1. (in Russian).

17. **Severson K. et al.** Elastic net with Monte Carlo sampling for data-based modeling in biopharmaceutical manufacturing facilities. *Computers & Chemical Engineering*. 2015, vol. 80, pp. 30-36.

18. **Nasehi Tehrani J. et al.** L1 regularization method in electrical impedance tomography by using the L1-curve (Pareto frontier curve). *Applied Mathematical Modelling*. 2012, vol. 36, no. 3, pp. 1095-1105.

19. **Tikhonov A.I.** Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*. 1963, no. 5, pp. 1035-1038.

20. **Kim S.-J. et al.** An Interior-Point Method for Large-Scale-Regularized Least Squares. *IEEE J. Sel. Top. Signal Process*. 2007, vol. 1, no. 4, pp. 606-617.

21. **Hwang E., Hong W.-T.** A multivariate HAR-RV model with heteroscedastic errors and its WLS estimation. *Economics Letters*. 2021, vol. 203, pp. 109855.

22. **Tellinghuisen J.** Chapter 10 Least Squares in Calibration: Weights, Nonlinearity, and Other Nuisances. *Methods in Enzymology. Academic Press*. 2009, vol. 454, pp. 259-285.

23. **Chen C. et al.** Fast iteratively reweighted least squares algorithms for analysis-based sparse reconstruction. *Medical Image Analysis*. 2018, vol. 49, pp. 141-152.

24. **Miller S., Startz R.** Feasible generalized least squares using support vector regression. *Economics Letters*. 2019, vol. 175, pp. 28-31.

25. **Liemohn M.W. et al.** RMSE is not enough: Guidelines to robust data-model comparisons for magnetospheric physics. *Journal of Atmospheric and Solar-Terrestrial Physics*. 2021, vol. 218, pp. 105624.

26. **Mueller S.** *Upgrading and Repairing PCs*. 19-th ed. / translate Tarabrova I.B. SPb: Izdatelskij dom «Vilyams». 2011, 1072 p. (in Russian).

27. **Bruce J., Spencer W.N., David T.W.** *Memory Systems Cache*, *DRAM, Disk*. USA: Elsevier, 2008. 982 p.