УДК 004.75

ПРОГНОЗИРОВАНИЕ НАГРУЗКИ С ИСПОЛЬЗОВАНИЕМ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОПРЕДЕЛЕНИЯ ПРИОРИТЕТА УЗЛА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

Т. Э. Зейналлы, аспирант, ассистент кафедры информатики и информационных технологий Московского Политехнического Университета, г. Москва, Россия;
orcid.org/0000-0001-6462-9607, e-mail: z.teymur.e@gmail.com
Д. Г. Демидов, к.т.н., доцент, декан факультета информационных технологий Московского политехнического университета, г. Москва, Россия;
orcid.org/0000-0002-2462-936X, e-mail: d.g.demidov@mospolytech.ru

Рассматриваются распределённые системы, состоящие из одноранговых узлов, итерационно выполняющие полезные нагрузки. Специфика рассматриваемых систем заключается в том, что приоритет рассчитывается в рамках одноранговых узлов – без узла координатора. Для них дается определение проблемы приоритизации узлов под выполнение конкретных задач, анализируются существующие научные исследования по данной тематике. ***Цель работы*** *– выработать метод прогнозирования приоритета узла для выполнения полезной нагрузки в системах с вышеописанными свойствами. Проводится анализ характеристик итерационных полезных нагрузок с целью определения и описания возможных сценариев приоритизации, выявления параметров и характеристик, оказывающих влияние на приоритет. Результатом анализа является способ вычисления приоритета, позволяющий системе адаптироваться к текущим условиям и рационально распределять нагрузку. Найденные характеристики позволяют адаптировать сценарии приоритизации для конкретных задач и повысить утилизацию ресурсов узла. В работе описывается экспериментальный стенд – система из трёх узлов, выполняющих итеративно полезную нагрузку. С него собираются данные для обучения нейронной сети. Описываются способы прогнозирования приоритета узла методами машинного обучения. Основным результатом работы является модель рекуррентной нейронной сети, осуществляющей прогнозирование приоритета узла для выполнения полезной нагрузки с учетом информации за предыдущие временные интервалы. В работе описываются входные и выходные данные модели. На вход модели передаются характеристики итерационного процесса и временные ряды исторических показателей нагрузки на узел. Выходным значением является показатель приоритета на следующий интервал времени. Оценивается точность модели прогнозирования. Применение предложенного метода помогает эффективным образом использовать ресурсы распределенных информационных систем. Приводятся варианты практического применения полученных в ходе работы результатов.*

Ключевые слова*: распределенная система, информационная система, программное обеспечение вычислительных комплексов, приоритизация узлов, рекуррентные нейронные сети, RNN, Long short-term memory LSTM, машинное обучение, прогнозирование.*

DOI*:* 10.21667/1995-4565-2024-90-54-66

Введение

В корпоративных информационных системах важным инструментом являются службы, которые осуществляют итерационные операции для выполнения полезной нагрузки. Для текущей работы содержимое полезной нагрузки не так важно, однако ее итеративное выполнение по определенному расписанию является обязательным условием. Расписания могут быть гибкими и зависеть от текущей нагрузки на систему. Для обеспечения надежности и отказоустойчивости описываемые системы горизонтально масштабируются, а выполнение полезной нагрузки при необходимости может координироваться через произвольное хранилище [1]. Системы, состоящие из таких служб, разворачиваются как на локальных серверах, так и в облачной среде. Подобные системы применяются для решения различных задач, таких как

автоматическая генерация отчетов, резервное копирование данных, мониторинг состояния системы и т. д.

В текущей работе под *worker процессом* понимается полезная нагрузка, которая выполняется итерационно, при этом необязательно на одном и том же узле кластера. Кластер выполняет параллельно произвольное количество различных полезных нагрузок, таких как разбор очередей, отправка уведомлений, обработка данных и т. д.

Для оптимизации работы таких служб применяются механизмы приоритизации [2]. Они позволяют эффективно распределять нагрузку по узлам кластера, отдавая выполнение работы наиболее подходящим узлам [3, 4].

В работе предлагается способ определения приоритета узла для выполнения конкретной полезной нагрузки. То есть некоторый узел будет более или менее приоритетным в зависимости от требований к ресурсам. Одним задачам важно процессорное время (они производят вычисления), другим – сетевой ресурс (занимаются скачиванием данных, например из облака), третьим – оперативная память. Предлагается способ вычисления приоритета с учётом данных прогноза нагрузки на узел. Прогноз основывается на специфике работы итерационных полезных нагрузок.

На текущий момент существует ряд исследований, описывающих использование методов прогнозирования, которые находят применение и в сфере корпоративных информационных систем. Эта задача известна как прогнозирование временных рядов [5, 6]. В актуальных исследованиях рассматривается прогнозирование отказов в распределённых системах [7], а также прогнозирование нагрузки на ресурсы узла, такие как оперативная память, процессор [8, 9]. Помимо этого, прогнозирование применяется для событий и показателей, имеющих какой-либо повторяющийся шаблон [10, 11]. В своих работах авторы используют различные подходы, включая нейронные сети и линейную регрессию.

Для задач прогнозирования высокую точность показывают рекуррентные нейронные сети на основе LSTM (Long short-term memory) [12]. Они имеют последовательную структуру повторяющихся модулей, как в обычных рекуррентных нейронных сетях. Однако из-за более сложной структуры самих модулей такие сети способны запоминать информацию в течение длительных периодов времени [13]. Это делает LSTM-сети полезными в задачах, где важно учитывать долгосрочные зависимости.

Среди исследований, базирующихся на использовании LSTM, есть работа Deepak Janardhanan и E. Barrett, которая сосредоточена на прогнозировании рабочей нагрузки процессора в центрах обработки данных с использованием рекуррентных нейронных сетей LSTM и сравнении их с моделями ARIMA [14]. Окончательная модель LSTM показала ошибку прогнозирования, меньшую, чем для модели ARIMA, что демонстрирует более высокую консистентность LSTM за счет их способности лучше обучаться на нелинейных данных.

Исследование Xiaoyong Tang посвящено проблеме прогнозирования рабочей нагрузки в крупномасштабных вычислительных системах с использованием улучшенной модели LSTM, которая разработана для повышения точности и скорости реализации прогнозов в реальном времени [15]. Автор предлагает двумерную структуру ячейки LSTM, а также параллельный алгоритм LSTM для достижения высокой точности прогнозирования на основе реальных данных Шанхайского суперкомпьютерного центра. Это исследование демонстрирует значительное улучшение точности прогнозов по сравнению с традиционными методами.

В работе J. Bi, Shuang Li, Haitao Yuan и Mengchu Zhou рассматривается интегрированный подход глубокого обучения для прогнозирования рабочей нагрузки и ресурсов в облачных системах [16]. Разработанный метод включает в себя модели сетей как двунаправленной, так и сеточной долгой кратковременной памяти (LSTM) для достижения высококачественного прогнозирования временных рядов рабочих нагрузок и ресурсов. Проведенное авторами экспериментальное сравнение показывает, что точность прогнозирования предложенного ими метода превосходит несколько широко используемых подходов.

Также среди недавних работ был представлен метод инкрементального онлайн-обучения для прогнозирования времени выполнения задач в научных параллельных программах в облаках [17]. Авторам удалось улучшить точность прогноза и снизить погрешности, используя данные мониторинга о потреблении ресурсов.

Отличительная особенность текущего исследования заключается в прогнозировании непосредственно приоритета узла. Для этого используются не только исторические данные о предыдущих итерациях, но и ресурсные потребности самих полезных нагрузок. Прогнозирование производится с учётом специфики кластера распределённых служб, которая выражается в расчете приоритета в рамках одноранговых узлов, т.е. без узла координатора (планировщика задач). Узлы не имеют информации о нагрузке на других узлах. Перечисленные аспекты позволяют обеспечить лучшую масштабируемость и делают метод наиболее подходящим для крупномасштабных и динамично изменяющихся распределённых систем.

Объектом исследования является кластер из служб, которые итерационно выполняют полезную нагрузку, которая может быть выполнена на любом из узлов распределённой системы. Узлы независимы друг от друга, но имеют равный доступ к общему хранилищу. Каждый из них имеет информацию о своих ресурсах, аппаратной конфигурации и конфигурации всех полезных нагрузок, а также имеет возможность проводить любые замеры показателей аппаратного обеспечения и полезных нагрузок. Каждый узел рассчитывает приоритет для каждой полезной нагрузки и заносит вычисленное значение в хранилище [2]. С учетом этих данных, полезная нагрузка будет выполнена на самом приоритетном для её выполнения узле.

Предметом исследования является способ вычисления приоритета узла для выполнения полезной нагрузки в данной системе. Приоритет – это рациональное число в диапазоне от 0 до 1. Чем меньше значение, тем операция приоритетнее. Операция со значением 0 имеет наивысший приоритет. Цель работы – выработать метод прогнозирования приоритета узла для выполнения полезной нагрузки в системах с вышеописанными свойствами. Задачи работы:

1. Провести анализ параметров, от которых зависит приоритет узла для выполнения конкретной полезной нагрузки, и однозначно определить критерии, по которым узел будет принимать решение о взятии на выполнение этой нагрузки.

2. Подготовить данные для обучения моделей.

3. Подготовить модель прогнозирования и оценить её точность.

Для обучения должны использоваться реальные данные, так как они отражают ситуации, возникающие на практике, что помогает сделать более достоверные выводы. Для сбора необходимо подготовить и развернуть кластер, выполняющий разнородные нагрузки, чтобы получить максимально разносторонний диапазон показателей.

Входными данными для модели являются набор показателей за последние N временных интервалов, параметры полезной нагрузки, выявленные в ходе анализа в первой задаче. Выходные данные представляют собой приоритет узла для выполнения полезной нагрузки, параметры которой были переданы. Модель должна базироваться на LSTM слоях. Данное решение обусловлено высокой точностью LSTM моделей для временных рядов, имеющих шаблонное поведение [13]. Таким образом, в модели на вход подается N временных интервалов, а на выходе получается числовое значение, представляющее собой приоритет узла. Этим данная модель отличается от LSTM моделей в классическом их понимании, где на выходе обычно получается значение прогнозируемого временного ряда. В текущей работе модель выполняет агрегирование информации из временных рядов, вычисляя итоговое значение приоритета. Отсюда возникает необходимость сравнения точности прогнозов данной модели с моделями, где на выходе производится прогнозирование значений временного ряда.

Методологической основой текущего исследования являются анализ, программная реализация и эксперимент. Проводится анализ сценариев использования системы с целью выявления параметров, влияющих на приоритет узла. Программная реализация полезных нагрузок производится с целью имитации нагрузки и последующего сбора данных для обучения модели. Эксперимент проводится для оценки точности моделей прогнозирования.

Кластер, с которого собирались данные, был развёрнут базе операционной системы linux с поддержкой виртуализации. Технические и программные характеристики устройства: CPU Intel i5-6200U (4) 2.8 GHz, RAM 12Gb, OS Manjaro Linux x86_64, Kernel 5.15.85-1. В качестве платформы контейнеризации был выбран Docker версии 20.10.22. В качестве хранилища ключ-значение был выбран etcd версии 3.3.8 с образом от quay.io. Для сборки показателей с узлов использовался telegraf версии 1.28.2. Функционирование кластера осуществляется по алгоритмам, описанным в работах [1-3].

Экспериментальная оценка моделей производится на языке Python (3.10.6), с основными пакетами: numpy (1.22.4), pandas (1.4.4), tensorflow (2.12.0), scikit-learn (1.1.2). Для возможности повторного воспроизведения полученных данных все материалы и модели выложены в Google Colaboratory (https://colab.research.google.com/drive/1UogiyaZepoFMvn8gabql6Fyso–NkarDec).

Также в работе описывается способ практического применения полученных результатов, программная реализация которого выполнена на платформе .NET 7.0 с использованием библиотеки ML.NET (2.0.1).

Анализ параметров, оказывающих влияние на приоритет

Необходимо определить, что должно учитываться при определении приоритета узла для выполнения полезной нагрузки. Первое, что кажется очевидным, – это учет времени, за которое полезная нагрузка выполнится на узле. Для ряда ситуаций такой подход оказывается полезным, так как различные узлы могут иметь аппаратное обеспечение с различной производительностью, и, таким образом, приоритет отдаётся более производительным узлам. Однако для систем с вышеописанными характеристиками длительность операции в ряде случаев может быть обусловлена количеством данных, обрабатываемых полезной нагрузкой в определённый момент времени. В зависимости от программной реализации одна и та же полезная нагрузка способна обрабатывать различное количество данных. Приоритизация по времени работы полезной нагрузки может привести к обратному эффекту.

В текущей работе для вычисления приоритета узла в определённый момент времени предлагается опираться на нагруженность узла в этот самый момент времени и на оценку потребности в ресурсах для конкретной полезной нагрузки.

Нагрузка на узел численно выражается по показателям метрик конкретных аппаратных ресурсов и представляет собой нормализованную на диапазон от 0 до 1 непрерывную величину.

Рассматриваемая в данной работе разновидность распределённых систем предназначена для параллельного выполнения ряда полезных нагрузок. Требования к ресурсам узла варьируются в зависимости от типа полезной нагрузки. Например, полезным нагрузкам, выполняющим ETL (Extract Transform Load), требуются процессорное время, сетевой ресурс и оперативная память [18]. Процессам отправки писем нужно относительно немного сетевого ресурса. Процессам обработки данных требуется в основном только CPU. Чтобы определить приоритет узла для выполнения конкретной полезной нагрузки, необходимо иметь информацию не только о текущей нагрузке, но и о том, какие ресурсы эта полезная нагрузка будет потреблять. В текущей работе эта оценка необходимости в ресурсах описывается числовыми значениями в диапазоне от 0 до 1, например: 0,3 CPU, 0,8 RAM и 0,7 NET.

Способ получения оценки необходимости в ресурсах может быть различным. Существует несколько подходов к её определению. Один из них – автоматическое вычисление системой средних показателей в процессе выполнения полезной нагрузки. Другой вариант – экспертная оценка, проводимая самим разработчиком полезной нагрузки. Разработчик может сформировать необходимость в ресурсах для полезной нагрузки, исходя из ожидаемого потребления ресурсов, основываясь на логике полезной нагрузки.

Ввиду того, что эти показатели выражают необходимость в том или ином ресурсе (метрике), далее по тексту они будут обозначаться как веса метрик для полезной нагрузки.

С учетом всего вышеперечисленного при вычислении приоритета должна учитываться как нагрузка (выраженная в показателях метрик), так и веса этих самых метрик.

$$p = max(m_1\omega_1, m_2\omega_2, ..., m_k\omega_k), \quad (1)$$

где m_k – это значение *k*-й метрики в определённый момент времени; ω_k – вес *k*-й метрики для полезной нагрузки; *p* – приоритет выполнения полезной нагрузки на узле.

Приоритет выполнения полезной нагрузки (1) на узле сочетает веса, отражающие важность ресурсов, и метрики, показывающие их текущую загрузку.

Например, предположим, что существует узел со следующими метриками: загрузка процессора (CPU) составляет $m_k = 0,89$, использование оперативной памяти (RAM) = 0,2, а использование сетевых ресурсов (NET) $m_3 = 0,17$. Если планируемая к выполнению полезная нагрузка требует значительных ресурсов по всем параметрам с весами $\omega_1 = 0,9$ для CPU, $\omega_2 = 0,9$ для RAM, $\omega_3 = 0,9$ для NET, то приоритет узла определяется как $p = max(0,89 \cdot 0,9; 0,2 \cdot 0,9; 0,17 \cdot 0,9) = 0,801$. В этом случае приоритет узла определяется значением загрузки процессора, так как это наиболее утилизируемый ресурс. Как было описано ранее, чем меньше числовой показатель приоритета, тем приоритетнее операция.

Если бы полезная нагрузка была менее требовательной к процессору, например с весом $\omega_1 = 0,2$, приоритет был бы равен $p = max(0,89 \cdot 0,2; 0,2 \cdot 0,9; 0,17 \cdot 0,9) = 0,18$. В данном случае решающим фактором при определении приоритета было значение оперативной памяти.

Таким образом, описанная формула вычисления приоритета позволяет избегать перегрузки наиболее важных (весомых) ресурсов.

Создание экспериментального стенда, замеры показателей (метрик)

Экспериментальная система состоит из трёх узлов приложения и трёх узлов хранилища. В зависимости от используемых ресурсов, выполняет три типа имитации полезной нагрузки:

1) на процессор от 10 % до 70 % на 4-10 секунд;
2) оперативную память, заполняя её от 1Gb до 4Gb на 1-5 секунд за каждый гигабайт;
3) сеть, скачивая данные на протяжении 10-20 секунд.

Каждая полезная нагрузка выполнялась с периодичностью, зависящей от того, насколько выбранное значение нагрузки было близко к пороговым значениям из вышеописанных диапазонов. Чем значение больше, тем чаще она выполнялась. Такой подход имитирует адаптивность возрастающей нагрузке, выраженной в увеличении объёма данных, требующем более частых итераций для их обработки.

Длительность имитации нагрузки выбиралась случайным образом в пределах диапазона, обозначенного выше.

Такой подход отражает реальные условия, когда объем обрабатываемых данных варьируется в зависимости от внешних факторов. Однако в реальных условиях объем данных изменяется не только случайно, но и в зависимости от других факторов, закономерностей или тенденций. Случайный выбор значений нагрузки представляет некоторую статистическую аппроксимацию этих изменений. В реальных сценариях объем обрабатываемых данных зависит от множества сложных факторов. Использование случайных значений нагрузки позволяет учесть эту вариабельность.

Для дальнейших расчётов потребуется измерять показатели аппаратного обеспечения. Это осуществляется с определённой выбранной периодичностью.

В качестве вариантов нагрузка на CPU и RAM может быть измерена в процентном соотношении, подобно методам, используемым графическими утилитами. Например, для операционной системы Windows можно использовать данные из performance counter, в то время как для Linux подходящим источником могут служить файлы /proc/stat и /proc/meminfo.

Измерение нагрузки на сеть в процентном соотношении затруднено из-за сложности в определении максимального объема отправляемого и принимаемого трафика. Вместо этого целесообразно сохранять количество отправленных или полученных данных в байтах.

В текущей работе для сбора этих показателей использовался telegraf. Собранные данные сохранялись в time series базу данных, расположенную на отдельном сервере, чтобы не оказывать существенного влияния на метрики.

Подготовка исходного набора данных

Основой для последующих расчётов являются данные, собранные на предыдущем этапе. Они могут представлять собой произвольные величины. Эти данные необходимо нормализировать на диапазон [0..1].

Некоторые показатели требуют предварительных операций перед нормализацией. Например, как было описано ранее, метрики сетевого трафика (отправленных и полученных байт) представляют собой возрастающую функцию. Для определения нагрузки на сеть в итоговом наборе данных потребуется посчитать скорость изменения этой функции (производную) или процентное изменение с предыдущим значением. И только потом нормализовать полученные значения.

Итого после нормализации имеется набор, состоящий из показателей трёх метрик – CPU (ЦП), RAM (ОЗУ), NET (сеть).

Необходимо, чтобы модель на выходе возвращала значение приоритета. Чтобы обучить модель формировать на выходном слое приоритет, необходимо в набор данных для каждой записи добавить параметры выполняемой полезной нагрузки, оказывающие влияние на приоритет (веса метрик) и сами значения приоритета.

Дополним набор данных. Для каждой метрики в наборе данных генерируются веса метрик для полезной нагрузки в диапазоне от 0 до 1 и записываются в соответствующую колонку веса метрики. В данном случае генерация весов метрик необходима для обучения модели на всём диапазоне входных значений весов метрик.

В получившемся наборе данных необходимо добавить и заполнить колонку приоритета в соответствии с приведенным ранее анализом, вычислив его для каждого временного интервала по формуле (1). Это необходимо сделать для того, чтобы обучаемая модель формировала приоритет в соответствии с ранее описанными принципами определения приоритета узла. Добавление вычисленного приоритета в качестве отдельной колонки обеспечит модели необходимый контекст для понимания взаимосвязи между нагрузкой на ресурсы (значениями метрик) и весами этих метрик.

Таблица 1 – Пример исходного набора данных
Table 1 – Example of a source data set

Time	Cpu	Ram	Net	Cpu (w)	Ram(w)	Net (w)	Priority
00:01	0,09	0,01	0,26	1,00	0,56	1,00	0,26
00:02	0,11	0,01	0,21	0,31	0,57	0,82	0,17
00:03	0,17	0,01	0,10	0,00	0,66	1,00	0,10
00:04	0,27	0,01	0,08	0,82	0,48	0,66	0,22

Собранные метрики, которые были нормализованы, визуализированы на рисунке 1.

Прогностическим моделям на вход подаётся N предыдущих значений временного ряда, на которых и будет основываться прогноз следующего значения. Это справедливо для одного временного ряда, в данной работе их три. Таким образом, на вход модели будет подаваться матрица размером Nx3.

Необходимо выяснить, какое значение может принимать N. Зачастую это значение подбирается эмпирически с учетом решаемой задачи, ресурсов и требуемого времени обучения модели. В данной работе было выбрано значение N, равное 20. Это значение позволяет

учесть возможные сложные зависимости и достаточно репрезентативно описывает исторические данные с учётом трёх временных рядов.

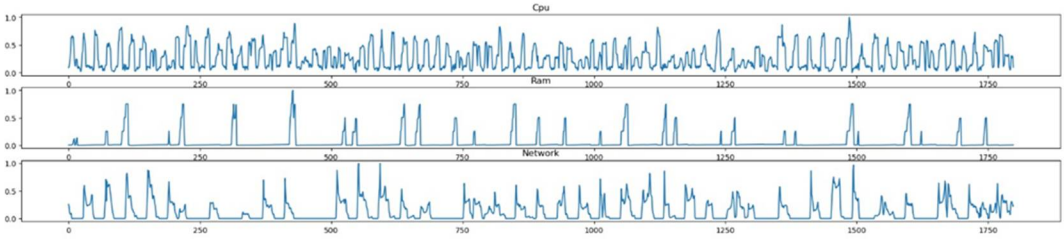

Рисунок 1 – Визуализация набора данных
Figure 1 – Dataset Visualization

Построение, обучение и оценка модели

Для обучения такой модели необходимо из таблицы исходных данных (таблица 1) сформировать обучающую и проверочную выборки. В данной работе с учетом правил кроссвалидации соотношение обучающей и проверочной выборок из исходных данных составляет 75 % / 25 %.

Выборка состоит из входных данных и ожидаемых данных на выходе. Входными данными одного элемента выборки являются показатели метрик за $(m_1, m_2, m_3)_i ... (m_1, m_2, m_3)_{i+N}$ и веса за следующий временной отрезок: $(\omega_1, \omega_2, \omega_3)_{i+N+1}$, так как с этими весами будет рассчитываться приоритет на выходе P_{i+N+1}. Другими словами, на вход подаётся матрица метрик за 20 временных отрезков и значения весов этих метрик за 21 временной отрезок, а на выходе ожидается спрогнозированный приоритет за 21-й временной отрезок. Этот подход формирования обучающей и проверочной выборки позволяет модели выстраивать закономерности того, как исторические данные по нагрузке оказывают влияние на показатель приоритета на следующий интервал времени.

В качестве функции потерь была выбрана среднеквадратичная ошибка (MSE – Mean Squared Error), которая измеряет среднюю квадратичную разницу между предсказанными значениями и истинными значениями. Большие ошибки влияют на функцию потерь значительно больше, чем маленькие, что заставляет модель акцентировать внимание на их устранении. Для задач, где важно предсказывать количественные значения, такой подход помогает улучшить точность модели.

Схематическая визуализация модели продемонстрирована на рисунке 2.

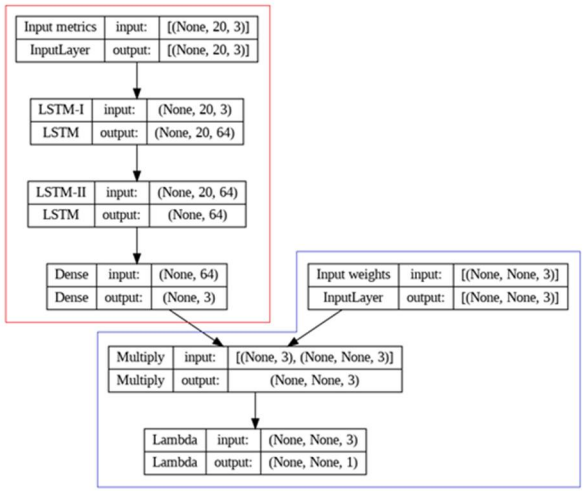

Рисунок 2 – Схематическая визуализация модели вычисления приоритета с прогнозированием
Figure 2 – Visualization of predictive priority calculation model

Модель имеет два входных слоя:

1) матрицы метрик;

2) передачи веса метрик для полезной нагрузки.

Красным цветом отмечена часть, отвечающая за прогнозирование, а синим цветом – за вычисление приоритета.

Модель состоит из двух частей, отвечающих за:

1) прогнозирование;

2) определение приоритета прогнозируемых метрик.

Часть прогнозирования состоит из:

1) входного слоя;

2) двух слоев LSTM;

3) полносвязного слоя.

Входной слой принимает тензор размерности (20, 3), где 20 – количество временных шагов, а 3 – количество метрик.

Первый LSTM слой имеет размер скрытого состояния 64, принимает входные данные размерности (20, 3) и выдает тензор размерности (20, 64). Второй LSTM слой также имеет размер скрытого состояния 64 и принимает на вход данные размерности (20, 64), а на выходе выдает тензор размерности (64). Этот слой обрабатывает выходные данные первого LSTM слоя, интегрируя информацию из предыдущих временных интервалов и обеспечивая более глубокое представление сложных и долгосрочных паттернов в данных.

Полносвязный слой (Dense) принимает входные данные размерности (64), преобразует их в выходной тензор размерности (3), а также выполняет линейное преобразование и служит для вычисления прогнозируемых значений метрик (m_1, m_2, m_3) на следующий временной интервал (N+1).

LSTM слои применяют рекуррентные нейронные сети с использованием механизмов забывания и обновления состояний, что позволяет моделировать долгосрочные зависимости во временных рядах. Эти слои используют специальные механизмы, такие как забывающие и входные гейты для регулирования потока информации. Это обеспечивает сохранение важных временных зависимостей и предотвращение исчезновения градиента при обучении.

Размерность скрытых состояний LSTM слоев и количество самих слоев подобраны эмпирически. Проведенные эксперименты показали, что при уменьшении размерности скрытых слоев или сокращении числа LSTM слоев показатели ошибок модели оставались без изменения, что указывает на их избыточность. Однако, выбранная конфигурация позволяет учесть потенциально сложные паттерны метрик.

Часть определения приоритета прогнозируемых метрик начинается со второго входного слоя, принимающий тензор весов размерности (3), и состоит:

1) из операции умножения (Multiply), которая принимает два тензора размерности (3);

2) первого тензора – выходного тензора полносвязного слоя из первой части модели;

3) второго тензора – входного тензора весов второго слоя.

В результате этой операции получается тензор размерности (3), который содержит произведение соответствующих элементов входных тензоров.

Затем получившийся тензор передается на слой Lambda, который вычисляет максимальное значение по определенной оси тензора. Этот слой принимает тензор размерности (3) и выдает тензор размерности (1).

После обучения данной модели были получены результаты, которые продемонстрированы на рисунке 3.

На рисунке 3 отмечены показатели приоритетов (ось Y) и порядковый номер временного интервала (ось X). Эти показатели имеют цветовую индикацию. Синим цветом отмечены показатели приоритета из исходного набора данных (таблица 1). Оранжевым цветом показаны приоритеты, которые модель формирует на обучающей выборке. Зелёным цветом показаны

приоритеты на проверочной выборке. Как видно на рисунке, спрогнозированные приоритеты сопоставимы с реальными значениями.

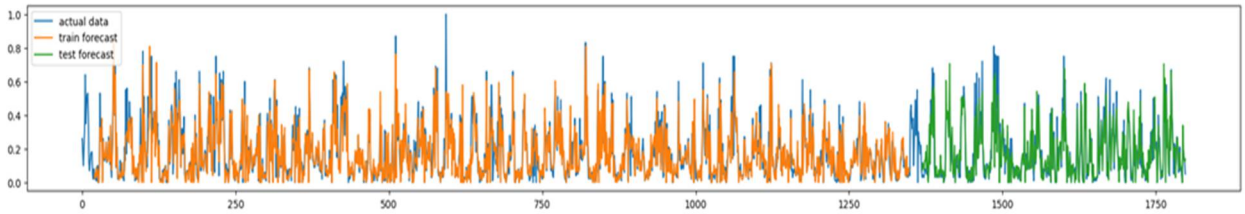

Рисунок 3 – Результаты прогнозирования приоритетов
Figure 3 – Priority prediction results

Для оценки прогностической точности модели применялись различные показатели: RMSE, MAE и R-squared. Корень из среднеквадратической ошибки (RMSE – Root Mean Squared Error) показывает среднюю величину ошибки в предсказаниях модели. Средняя абсолютная ошибка (MAE – Mean Absolute Error), в отличие от RMSE, дает равный вес всем ошибкам независимо от их величины. Коэффициент детерминации (R^2) измеряет, насколько хорошо будущие выборки будут предсказываться моделью (чем ближе к 1, тем лучше).

Таблица 2 – Показатели для обучающей и проверочной выборок
Table 2 – Metrics for training and validation samples

	RMSE	MAE	R^2
Обучающая выборка	0,08	0,05	0,77
Проверочная выборка	0,09	0,06	0,70

Данная модель помимо прогнозирования выполняет также расчет приоритета. Чтобы убедиться в точности модели, стоит отдельно оценить точность её первой части, которая отвечает за прогнозирование, и сравнить полученные результаты с текущими.

Если рассматривать только первую часть модели, то выходным слоем будет полносвязный слой с показателями метрик на следующий интервал времени.

По итогам обучения, такая модель показала следующие результаты.

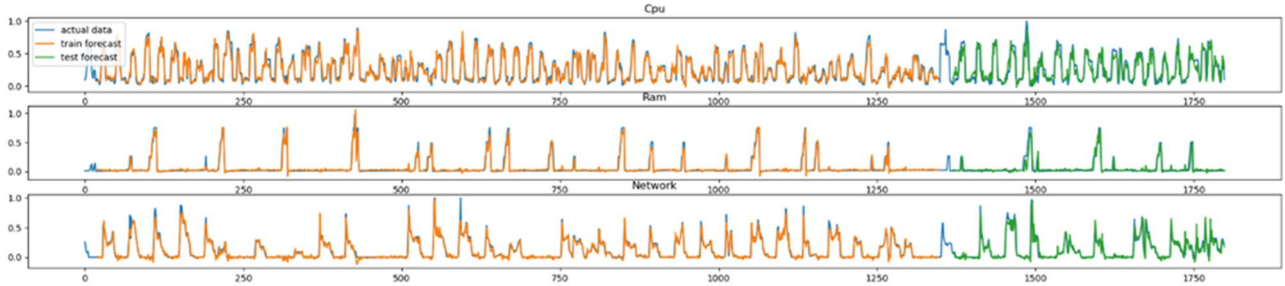

Рисунок 4 – Результаты прогнозирования метрик
Figure 4 – Metric prediction results

Цветовая индикация соответствует рисунку 3. На представленном графике визуализированы спрогнозированные показатели метрик. Как и на предыдущем рисунке, спрогнозированные данные сопоставимы с реальными.

Таблица 3 – Показатели для обучающей и проверочной выборок для первой части модели
Table 3 – Metrics for Training and Validation Samples for the First Part of the Model

	RMSE	MAE	R^2
Обучающая выборка	0,07	0,05	0,83
Проверочная выборка	0,09	0,06	0,77

Полученные показатели ошибок позволяют сделать вывод о том, что дополнительные слои не ухудшают результаты.

Полученные результаты имеют низкий показатель ошибки. Высокая точность модели обусловлена спецификой задачи прогнозирования итерационных процессов в рассматриваемых системах. Данные для обучения модели были получены из реальных замеров на операционной системе и отражают типичные повторяющиеся шаблоны нагрузки, характерные для таких систем. Итерационные процессы благодаря своей рекуррентности хорошо поддаются прогнозированию, поскольку для них проще выявить закономерности в обучающих данных.

Таким образом, представленная модель отвечает практическим требованиям и может быть использована для прогнозирования приоритета узла в системах с рассматриваемой спецификой. Однако в практических условиях потребуется периодическое дообучение модели с учётом новых данных, характерных для конкретной системы.

Способ практического использования

Для практического использования результатов данной работы понадобится дообучать или переобучать модель с появлением новых данных. Вопрос частоты переобучения индивидуален для каждой системы. Однако не следует это делать на узлах, выполняющих полезные нагрузки. Для переобучения следует выделить отдельный узел (ML-узел), который будет заниматься только обучением моделей для всех остальных узлов системы (worker-узлов). Worker-узлы обладают информацией о том, какие веса имеет каждая полезная нагрузка, выполняющаяся на них. Worker-узлы могут осуществлять замеры своих метрик с определённой периодичностью. Для определения приоритета для конкретной полезной нагрузки узел использует N своих метрик, веса полезной нагрузки и подаёт их на вход модели для получения приоритета полезной нагрузки для текущего узла системы.

Чтобы обучать новую модель, worker-узел отправляет собранные метрики на ML узел, а он уже возвращает обученную модель по актуальным метрикам.

Альтернативный вариант – в инфраструктуре уже используются средства сбора метрик с последующим сохранением в базу данных временных рядов. В таком случае эти базы следует сделать источником данных для ML сервера, а worker-узлы будут только скачивать обученные модели.

Одной из потенциальных стратегий для разработки сервера машинного обучения (ML) является использование языка программирования Python, поскольку он обладает обширным набором инструментов и библиотек, специализированных для задач машинного обучения. Для использования обученных ML моделей в логике worker-узлов существуют специализированные библиотеки и инструменты, обеспечивающие их использование на различных языках программирования. Одним из таких решений является формат ONNX (Open Neural Network Exchange), который активно развивается как универсальный стандарт для переноса моделей между различными платформами и фреймворками. Он позволяет интегрировать обученные модели в приложения с минимальными усилиями, сохраняя совместимость с популярными инструментами и экосистемами [19]. Представленная в текущей работе практическая реализация для использования моделей была апробирована с использованием ONNX моделей на .NET. ML сервер был написан на python с tensorflow.

Заключение

Результаты текущего исследования вносят вклад в область прогнозирования приоритетов узлов для итерационного выполнения полезных нагрузок в распределенных системах и позволяют сделать следующие выводы.

Учет различных требований к ресурсам узлов для полезных нагрузок при вычислении приоритета позволит более рационально распределять нагрузку на узлы, что приведет к улучшению производительности и повысит адаптивность системы к изменяющейся нагрузке.

Разработанная модель прогнозирования, основанная на LSTM слоях, показала:

1) высокую точность, о чем свидетельствуют низкие значения среднеквадратичной ошибки (RMSE) и средней абсолютной ошибки (MAE) для обеих выборок;

2) соответствие обучающим данным, на что указывает полученный коэффициент детерминации (R-squared);

3) высокую прогностическую способность на новых данных.

В работе описано как предложенная модель на практике может быть интегрирована в системы без существенных изменений в их архитектуре. Однако, для практического применения предполагается переобучать (или доучивать) модель под данные системы для её адаптации к изменяющимся паттернам временных рядов метрик.

Предложенная модель обладает хорошей обобщающей способностью и может применяться на практике для прогнозирования приоритетов узла. Полученные результаты будут полезными для разработчиков и инженеров, специализирующихся в области распределенных систем.

Библиографический список

1. **Zeynally T., Demidov D.** Fault Tolerance of Distributed Worker Processes in Corporate Information Systems and Technologies // Communications in Computer and Information Science, 2022.

2. **Zeynally T., Demidov D., Dimitrov L.** Prioritization of Distributed Worker Processes Based on Etcd Locks // Communications in Computer and Information Science, 2022.

3. **Zeynally T., Demidov D.** Evaluation of the Efficiency of Fault Tolerance Algorithms for Distributed Peer-To-Peer Worker Processes Connected Through a Key-Value Store // Communications in Computer and Information Science, vol. 1821. Springer, 2023: Материалы конференции Information Technologies and Intelligent Decision Making Systems (ITIDMS), Москва, Московский Политехнический Университет, 2022.

4. **Зейналлы Т. Э.** Сравнительный анализ решений на базе .NET для итерационного выполнения задач в распределенных отказоустойчивых системах // International Journal of Open Information Technologies. 2024. № 6.

5. **Kaufmann M.** Data Science Concepts and Practice Time Series Forecasting, 2018. 395 p. ISBN: 9780128147610.

6. **Bisht D.C.S., Ram M.** Computational Intelligence-based Time Series Analysis. River Publishers, India, 2022.

7. **Сай Ван Квонг, Щербаков М.В.** Прогнозирование отказов сложных многообъектных систем на основе комбинации нейросетей: пути повышения точности прогнозирования // Прикаспийский журнал: управление и высокие технологии. 2020. № 1 (49).

8. **Тузов А.В.** Исследование возможности использования линейной регрессии для предсказания расхода памяти в высоконагруженной информационной системе // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2018. № 3.

9. **Кучерова К.Н.** Прогнозирование ресурсов облачных сервисов на основе мониторинговой системы с открытым кодом // Труды учебных заведений связи. 2020. № 3.

10. **Чирков И.А., Дунаев М.Е.** Исследование возможностей нейронных сетей для прогнозирования показателей функционирования брокеров сообщений технологических платформ // International Journal of Open Information Technologies. 2021. № 8.

11. **Осипов В.Ю., Милосердов Д.И.** Нейросетевое прогнозирование событий для роботов с непрерывным обучением // Информационно-управляющие системы. 2020. № 5 (108).

12. **Yang B., Sun S., Li J., Lin X., Tian Y.** Traffic flow prediction using LSTM with feature enhancement // Neurocomputing, 2019, vol. 332, pp. 320-327. DOI: 10.1016/j.neucom.2018.12.016.

13. **Ekiz D., Can Y.S., Ersoy C.** Long Short-Term Memory Network Based Unobtrusive Workload Monitoring With Consumer Grade Smartwatches // IEEE Transactions on Affective Computing. 2023. Vol. 14, no. 2, pp. 895-905.

14. **Janardhanan D., Barrett E.** CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models // 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST). 2017, pp. 55-60.

15. **Tang X.** Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network // IEEE Access, 2019, vol. 7, pp. 40525-40533.

16. **Bi J., Li S., Yuan H., Zhou M.** Integrated deep learning method for workload and resource prediction in cloud systems // Neurocomputing, 2020, vol. 424, pp. 35-48.

17. **Hilman M.H., Rodriguez M.A., Buyya R.** Task runtime prediction in scientific workflows using an online incremental learning approach // 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC). Zurich, Switzerland. IEEE, 2018, pp. 93-102.

18. **Vijayalakshmi M., Minu R.I.** Incremental Load Processing on ETL System through Cloud // International Conference for Advancement in Technology (2022 ICONAT). Goa, India, 2022.

19. **Jajal P., Jiang W., Tewari A., Woo J., Lu Y.-H., Thiruvathukal G.K., Davis J.C.** Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem, 2023.

UDC 004.75

LOAD FORECASTING USING RECURRENT NEURAL NETWORKS TO DETERMINE NODE PRIORITY IN A DISTRIBUTED SYSTEM

T. E. Zeynally, postgraduate student, assistant at the Department of Computer Science and Information Technologies, Moscow Polytechnic University, Moscow, Russia;
orcid.org/0000-0001-6462-9607, e-mail: z.teymur.e@gmail.com
D. G. Demidov, Ph.D. (Tech.) Dean of IT Faculty, Moscow Polytechnic University, Moscow, Russia;
orcid.org/0000-0002-2462-936X, e-mail: d.g.demidov@mospolytech.ru

This paper examines distributed systems composed of peer nodes that iteratively perform useful workloads. The specific feature of the systems considered is that priority is calculated among peer nodes – without a coordinator node. The problem of prioritizing nodes for executing specific tasks is defined for these systems, and existing scientific research on this topic is analyzed. ***The aim of the work*** *is to develop a method for predicting the priority of a node for executing a payload in systems with the above-described properties. An analysis of iterative workloads characteristics is conducted to determine and describe possible prioritization scenarios and to identify parameters and characteristics that influence priority. The result of the analysis is a method for calculating priority that enables the system to adapt to current conditions and distribute the load rationally. The identified characteristics allow adaptation of prioritization scenarios for specific tasks and improvement of node resource utilization. The paper describes experimental setup – a system of three nodes iteratively performing useful workloads. The data collected from this setup is used to train a neural network. Methods for predicting node priority using machine learning techniques are described. The main result of the work is a recurrent neural network model that predicts the priority of a node for executing useful workloads, based on the information from previous time intervals. The paper describes model input and output data. The inputs to the model include characteristics of iterative process and time series of historical load indicators on a node. The output value is a priority indicator for next time interval. The accuracy of forecasting model is evaluated. The application of the proposed method helps to effectively utilize the resources of distributed information systems. Practical applications of the results obtained in the course of the work are presented.*

Keywords*: distributed system, information system, computer software, node prioritization, RNN, LSTM, machine learning, forecasting.*

DOI: 10.21667/1995-4565-2024-90-54-66

References

1. **Zeynally T., Demidov D.** Fault Tolerance of Distributed Worker Processes in Corporate Information Systems and Technologies. *Communications in Computer and Information Science*, 2022.

2**. Zeynally T., Demidov D., Dimitrov L.** Prioritization of Distributed Worker Processes Based on Etcd Locks. *Communications in Computer and Information Science*, 2022.

3. **Zeynally T., Demidov D.** Evaluation of the Efficiency of Fault Tolerance Algorithms for Distributed Peer-To-Peer Worker Processes Connected Through a Key-Value Store. *Communications in Computer and Information Science*, vol. 1821. Springer, 2023: *Materialy konferentsii Information Technologies and Intelligent Decision Making Systems (ITIDMS)*, Moskva, Moskovskii Politekhnicheskii Universitet, 2022. (*Proceedings of the conference Information Technologies and Intelligent Decision Making Systems (ITIDMS)*, Moscow, Moscow Polytechnic University, 2022).

4. **Zeynally T.E.** Sravnitel'nyy analiz resheniy na baze .NET dlya iteratsionnogo vypolneniya zadach v raspredelennykh otkazoustoychivykh sistemakh (Comparative Analysis of .NET-Based Solutions for Iterative Task Execution in Distributed Fault-Tolerant Systems). *International Journal of Open Information Technologies*. 2024, no. 6. (in Russian).

5. **Kaufmann M.** Data Science Concepts and Practice Time Series Forecasting, 2018. 395 p. ISBN 978--0-12-814761-0.

6. **Bisht D.C.S., Ram M.** Computational Intelligence-based Time Series Analysis. River Publishers, India, 2022.

7. **Sai Van Kvong, Shcherbakov M.V**. Prognozirovanie otkazov slozhnykh mnogoobektnykh sistem na osnove kombinatsii neirosetei: puti povysheniia tochnosti prognozirovaniia (Failure prediction of complex multi-object systems based on the combination of neural networks: ways to improve prediction accuracy). *Prikaspiskii zhurnal: upravlenie i vysokie tekhnologii*. 2020, no. 1 (49). (in Russian).

8. **Tuzov A.V.** Issledovanie vozmozhnosti ispol'zovaniya linejnoj regressii dlya predskazaniya raskhoda pamyati v vysokonagruzhennoj informacionnoj sisteme. *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta*. Seriya: Komp'yuternye tekhnologii, upravlenie, radioelektronika. 2018. no. 3. (in Russian).

9. **Kucherova K.N.** Prognozirovanie resursov oblachnykh servisov na osnove monitoringovoi sistemy s otkrytym kodom (Forecasting cloud services resources based on an open-source monitoring system). *Trudy uchebnykh zavedenii sviazi*. 2020, no. 3. (in Russian).

10. **Chirkov I.A., Dunaev M.E.** Issledovanie vozmozhnostei neironnykh setei dlia prognozirovaniia pokazatelei funktsionirovaniia brokerov soobshchenii tekhnologicheskikh platform (Investigation of neural networks capabilities for predicting the performance indicators of message brokers of technological platforms). *International Journal of Open Information Technologies*. 2021, no. 8. (in Russian).

11. **Osipov V.Yu., Miloserdov D.I.** Neirosetevoe prognozirovanie sobytii dlia robotov s nepreryvnym obucheniem (Neural network event forecasting for robots with continuous learning). *Informatsionno-upravliaiushchie sistemy*. 2020, no. 5 (108). (in Russian).

12. **Yang B., Sun S., Li J., Lin X., Tian Y.** Traffic flow prediction using LSTM with feature enhancement. *Neurocomputing*, 2019, vol. 332, pp. 320-327. DOI: 10.1016/j.neucom.2018.12.016.

13. **Ekiz D., Can Y.S., Ersoy C.** Long Short-Term Memory Network Based Unobtrusive Workload Monitoring With Consumer Grade Smartwatches. *IEEE Transactions on Affective Computing*. 2023, vol. 14, no. 2, pp. 895-905.

14**. Janardhanan D., Barrett E.** CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. 2017. *12th International Conference for Internet Technology and Secured Transactions (ICITST)*. 2017, pp. 55-60.

15. **Tang X.** Large-Scale Computing Systems Workload Prediction Using Parallel Improved LSTM Neural Network. *IEEE Access*, 2019, vol. 7, pp. 40525-40533.

16. **Bi J., Li S., Yuan H., Zhou M.** Integrated deep learning method for workload and resource prediction in cloud systems. *Neurocomputing*, 2020, vol. 424, pp. 35-48.

17. **Hilman M.H., Rodriguez M.A., Buyya R.** Task runtime prediction in scientific workflows using an online incremental learning approach. 2018. *IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*. Zurich, Switzerland. IEEE, 2018, pp. 93-102.

18. **Vijayalakshmi M., Minu R.I.** Incremental Load Processing on ETL System through Cloud. *International Conference for Advancement in Technology (2022 ICONAT)*. Goa, India, 2022.

19. **Jajal P., Jiang W., Tewari A., Woo J., Lu Y.-H., Thiruvathukal G.K., Davis J.C.** Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem, 2023.