УДК 621.391 **DOI 10.52928/2070-1624-2024-42-1-18-25**

ОПЕРАТИВНОЕ УПРАВЛЕНИЕ КРИТИЧЕСКИМИ IT-СИСТЕМАМИ НА ОСНОВЕ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

А. А. СТАРОВОЙТОВ, д-р техн. наук, проф. В. В. КРАСНОПРОШИН
(Белорусский государственный университет, Минск)

A. Starovoytov ORCID https://orcid.org/0009-0001-6531-4225,
V. Krasnoproshin ORCID https://orcid.org/0000-0002-9463-4869

Исследуется актуальная прикладная проблема, связанная с оперативным управлением критическими информационными сервисами. Предложен оригинальный подход, основанный на нейросетевом прогнозировании, в рамках которого разработан метод динамической локальной аппроксимации нейросетевыми моделями. Изложены принципы построения и реализации алгоритма функционирования (в условиях неопределенности профиля внешней нагрузки) комбинированной проактивной системы управления вычислительными ресурсами. Проведены эксперименты, которые подтвердили эффективность метода и подхода в целом.

Ключевые слова: *принятие оперативных решений, критическая информационная система, проактивное управление, неопределенность внешней нагрузки, нейронные сети, нейромодуль.*

Введение. Поддержка критических IT-сервисов и систем (банковских, телекоммуникационных, промышленных) в рабочем состоянии (с гарантированным объемом вычислительных ресурсов) является актуальной проблемой современного цифрового общества.

Неопределенность внешней нагрузки, отказы вычислительного оборудования приводят к сбоям в работе и деградации производительности критических IT-систем. В результате этого теряется оперативность обработки информации и проведения банковских и других операций. Все это, в свою очередь, может иметь серьезные последствия (финансовые потери, крупные аварии и т. п.).

Принятие оперативных решений по управлению критическими IT-сервисами позволяет уменьшить (или не допустить) возникновение негативных последствий. Человеческий фактор часто является причиной снижения оперативности, поэтому активно развиваются автоматизированные подходы, направленные на повышение уровня оперативности и проактивности в управлении.

Одним из перспективных подходов к решению проблемы является использование интеллектуальных систем, реализующих схему: сбор данных – построение прогностической модели – проактивное принятие решений – выполнение управляющих действий.

Рядом авторов разработаны системы управления с использованием нейросетевых моделей, способствующие эффективному принятию решений [1–4]. Однако в данных системах для определенных типов внешней нагрузки обучается только одна нейросетевая модель. Предполагается, что эта модель будет успешно прогнозировать значения необходимых параметров и для других (связанных с неопределённостью) типов нагрузки. Выборки для обучения таких моделей готовятся заранее, они, как правило, содержат длительные по времени данные с большим количеством элементов. Соответственно, для эффективного обучения модели (за допустимое время) требуется большое количество высокопроизводительных ресурсов и времени.

В работе предлагается подход, основанный на предположении, что управляемая IT-система может находиться в различных (по объему вычислительных ресурсов) состояниях. Для каждого состояния (в течение времени его существования), может быть создана и обучена нейросетевая модель, которая способна предсказывать среднее значение утилизации %CPU по вычислительным модулям. На основании этого значения может быть принято управляющее решение, изменяющее состояние системы.

Модельная система критического IT-сервиса. Для проведения экспериментов авторами была разработана модельная система, концептуально соответствующая модели критического информационного сервиса [5]. Система состоит из набора вычислительных модулей, на которых функционируют инстансы прикладного программного обеспечения, между которыми осуществляется балансировка внешних запросов. Для генерации запросов используется система, позволяющая задавать нагрузку и получать статистику по обработанным запросам и временам отклика. Модельная система поддерживает механизм масштабирования (изменения состояния). Основные блоки модельной системы представлены на рисунке 1.

Вычислительных модули, на которых функционирует прикладной сервис, реализованы с помощью Docker Compose. Данное решение позволяет создавать вычислительные модули (Docker-контейнеры), управлять ими, собирать различные метрики утилизации. Доступны библиотеки (Docker SDK for Python) для работы с Docker API Engine. Балансировка нагрузки по контейнерам с прикладным web-приложением осуществляется встроенными в Docker Compose средствами по имени сервиса. Для управления прикладным сервисом и ресурсами используется подход Infrastructure as a Code (IaC). Конфигурация описывается в виде файла в формате yaml.

Ввиду экономии ресурсов, стабильности работы, простоты настройки и эксплуатации в качестве прикладного web-приложения было решено реализовать микросервис, основанный на популярном высокопроизводительном минималистичном web-фреймворке echo.labstack, написанном на высокоуровневом языке Go.

В качестве генератора нагрузки используется Locust: написан на Python, позволяет описывать профили нагрузки в виде классов, поддерживает распределенную конфигурацию инстансов, позволяет задавать профиль нагрузки в виде функции или заранее подготовленных данных.

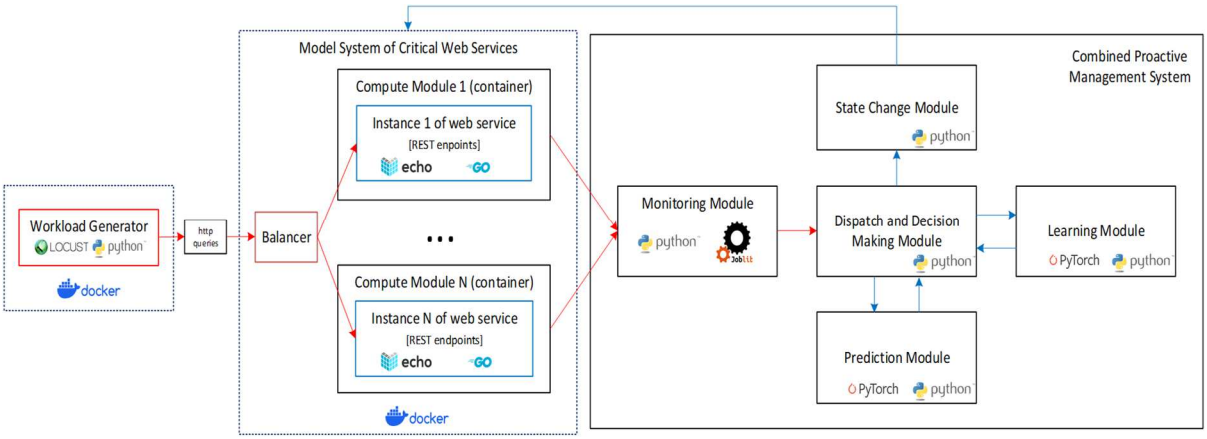

Рисунок 1. – Структурная блок-схема комбинированной системы управления с нейромодулем

Комбинирования система проактивного управления (КСПУ). Архитектурно КСПУ состоит из 5 основных блоков-модулей (см. рисунок 1):

1. Блок мониторинга – отвечает за сбор метрик производительности вычислительных модулей системы и добавление новых метрик (при изменении состояния системы). Метрики собираются каждые 2 с.
2. Блок изменения состояния – отвечает за отправку управляющих команд, которые изменяют состояние управляемой системы, и контроль корректности изменения состояния.
3. Блок диспетчеризации и принятия решений.
4. Модуль обучения – создает модели нейросетей для набора исторических данных для различных состояний управляемой системы.
5. Модуль прогнозирования – выполняет прогнозы параметров утилизации с определенной заблаговременностью для различных состояний управляемой системы.

Блок диспетчеризации и принятия решений является центральным, в него попадает информация из блока мониторинга. По последним переданным данным производится расчет средних значений по набору вычислительных модулей. Если средние значения по вычислительным модулям превышают пороговые, то принимается решение по изменению состояния управляемой системы и отправляется команда в блок изменения состояния (реактивная компонента).

Параллельно с этим процессом в блоке диспетчеризации и принятия решений происходит накопление исторических данных по вычислительным модулям, которые передаются в модуль обучения. При достижении определенного объема данных происходит их передача в модуль, в котором выполняется обучение нейросети на этом наборе данных. После получения рабочей модели блок обучения передает информацию о модели в центральный блок, который берет полученные данные из блока мониторинга и отправляет их и код модели нейросети в блок прогнозирования. Данные механизмы неблокирующего взаимодействия между процессами реализованы с помощью библиотеки multiprocessing [Python]. Более детальное описание технологии и программного алгоритма работы КСПУ представлено в работах [6; 7].

Прогнозирование. Состояние критической системы связано с объемом вычислительных ресурсов, который требуется для нормального функционирования системы для конкретной нагрузки. Состояние определяется количеством используемых вычислительных модулей. Данные для построения прогностической модели берутся из локальной временной области, соответствующей конкретному состоянию системы. В качестве метрики используется среднее значение %CPU по набору вычислительных модулей.

Время жизни конкретного состояния зависит от характера нагрузки. Переход в другое состояние определяется уровнями измеряемой метрики (уровнями переходов). Задаются максимальный уровень средней утилизации %CPU по вычислительным модулям, при котором система автоматически переходит в новое состояние с большим количеством ресурсов, и минимальный уровень, при котором система переходит в состояние с меньшим количеством ресурсов.

Локальные временные области для различных состояний могут иметь различное количество отсчетов. В это количество входят отсчеты, необходимые для создания прогностической модели. На подстройку параметров модели и обучение также требуется определенное время.

Из-за неопределенности внешнего воздействия допустимы резкие скачки нагрузки за характерный временной интервал, который меньше суммарного времени создания модели и подготовки прогноза. В этом случае система изменяет состояние на основе реактивного управления.

Существуют временные ограничения на процесс создания и обучения модели. Желательно, чтобы проактивная компонента успевала построить адекватную модель и подготовить прогноз, на основе которого будет выполнен переход в новое состояние раньше, чем это произойдет на основе реактивного управления.

В статье [7] авторами для построения прогноза в локальной временной области была сформулирована следующая задача:

– пусть задана критическая система, которая может находиться в конечном множестве состояний $S = \{S_1, S_2, \dots S_n\}$, которые определяются воздействием на систему и уровнями переходов. В каждом состоянии *i* система генерирует дискретный сигнал в виде короткого временного ряда $X_t^{S_i}$ (или набора рядов). Необходимо по части этого сигнала $\tilde{X}_t^{S_i}$ построить предиктор в виде нейронной сети, который прогнозирует переход системы в новое состояние S_{i+1}.

Для решения данной задачи был разработан метод динамической локальной аппроксимации нейросетевыми моделями (DLANN). Суть данного метода заключается в том, что в процессе работы системы для каждого её состояния строится простая нейросетевая модель (с одним скрытым и одним выходным слоем), которая учится на части данных локальной временной области. Данный метод послужил основной для разработки КСПУ.

Для создания и обучения моделей используется библиотека Pytorch [Python]. Процесс обучения должен занимать мало времени, поэтому используется небольшое количество эпох обучения и простая нейронная сеть с одним скрытым слоем (nn.Linear). Функция активации – ReLU, для регуляризации используется отключение части нейронов (dropout). Функция потерь – nn.MSELoss(). Метод оптимизации – torch.optim.Adam, скорость обучения 0,002.

Время обучения нейронной сети с указанной архитектурой для набора данных составляет ~ 2 с. Скорость выполнения предсказания с учетом заблаговременности в 40 отсчетов $\ll 1$ с.

Результаты экспериментов. В качестве примера рассмотрим результаты работы управляющей системы при постепенно возрастающей нагрузке, достигающей 600 пользователей, выполняющих различные запросы примерно за 0,5 ч. В процессе эксперимента система меняла свое состояние 6 раз, при этом проактивные изменения происходили 4 раза (переходы S0→S1, S2→S3, S4→S5, S5→S6), реактивные изменения происходили 2 раза (переходы S1→S2, S3→S4). Графики с исходными данными для прогноза и прогнозом для состояний S0, S1, S2, S3, S4, S5 представлены на рисунках 2–7.

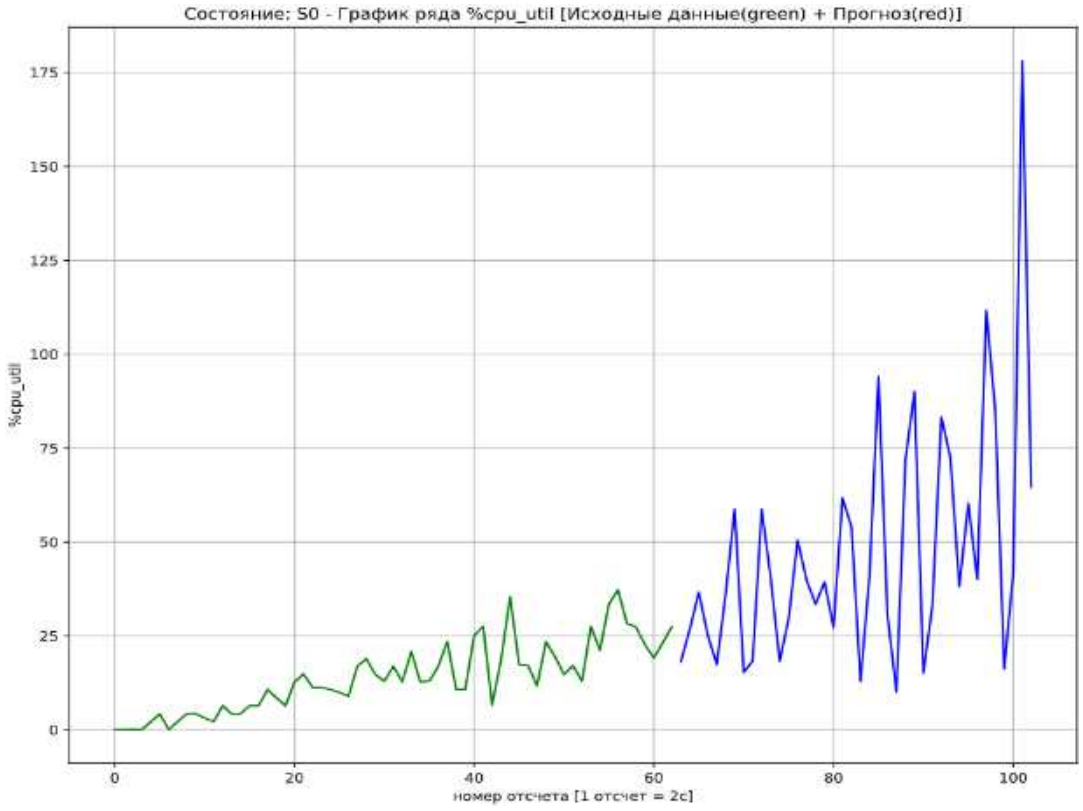

Рисунок 2. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S0

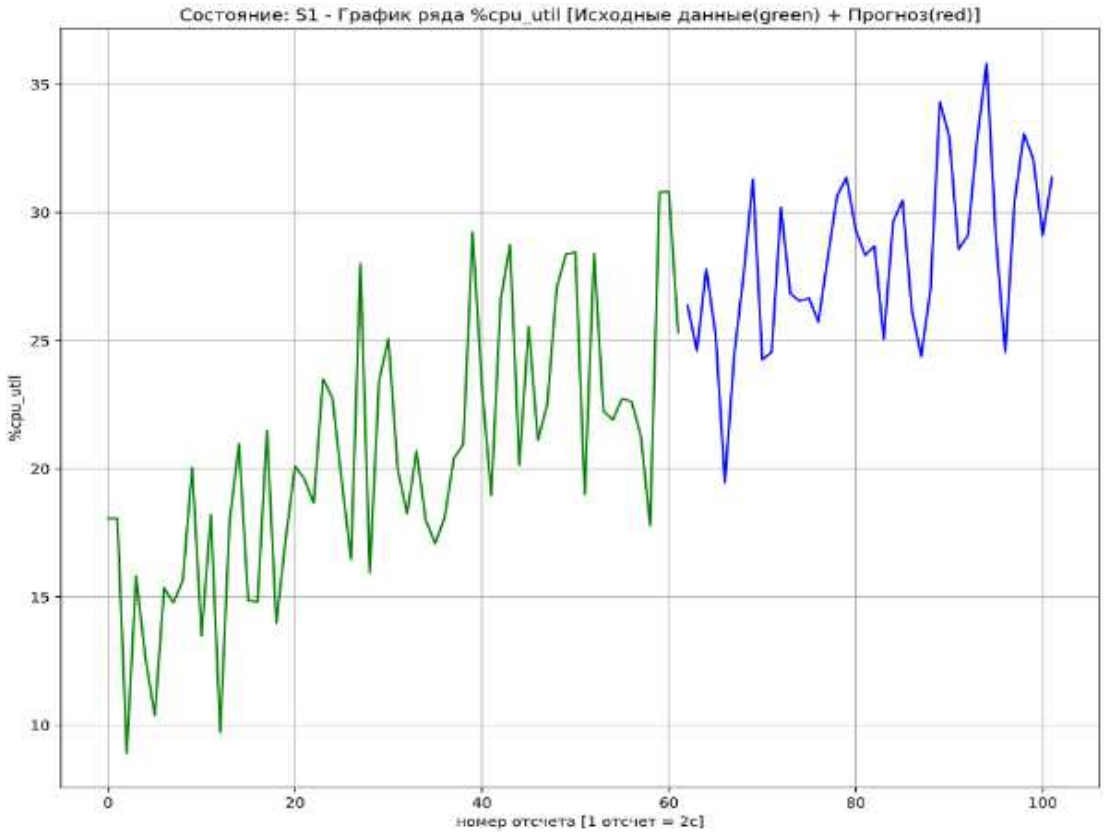

Рисунок 3. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S1

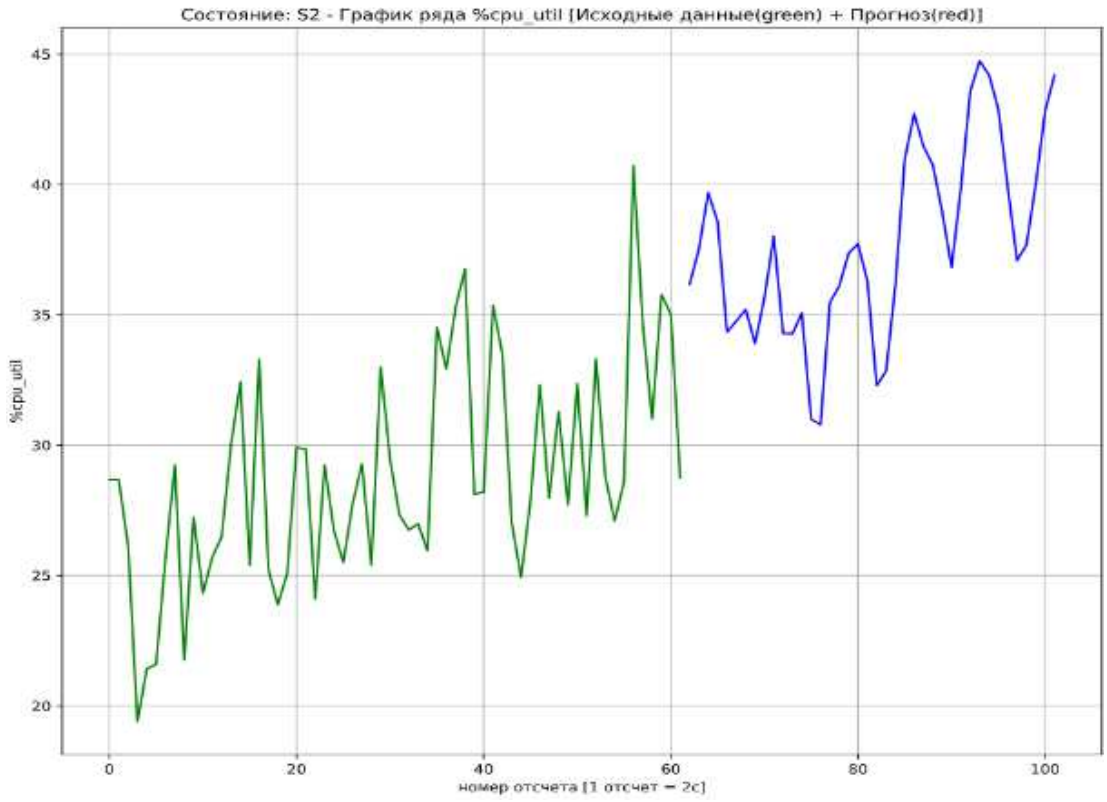

Рисунок 4. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S2

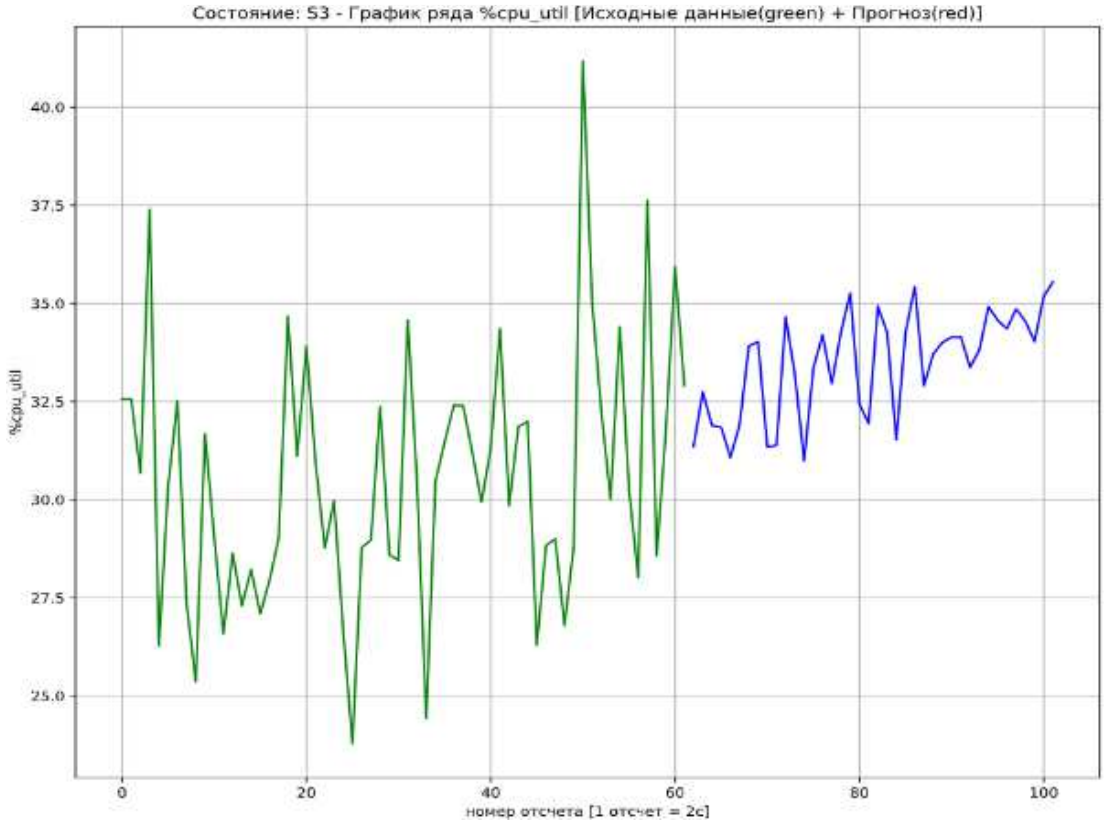

Рисунок 5. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S3

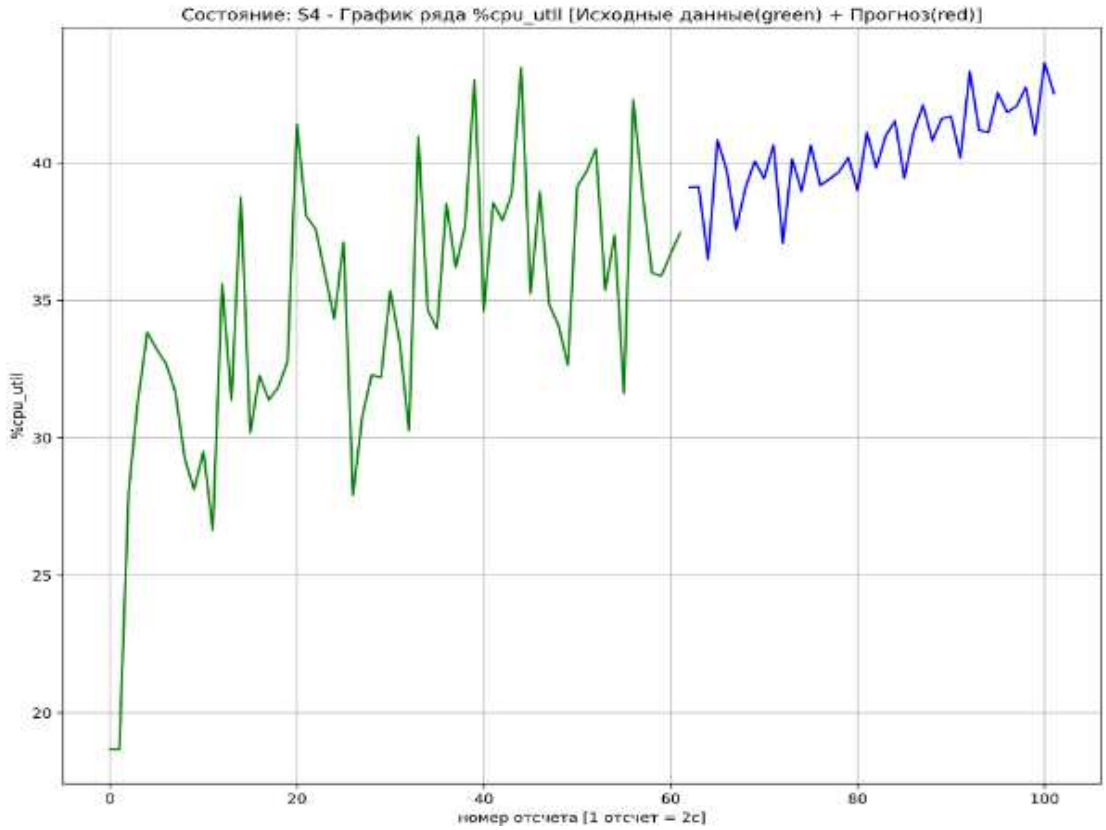

Рисунок 6. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S4

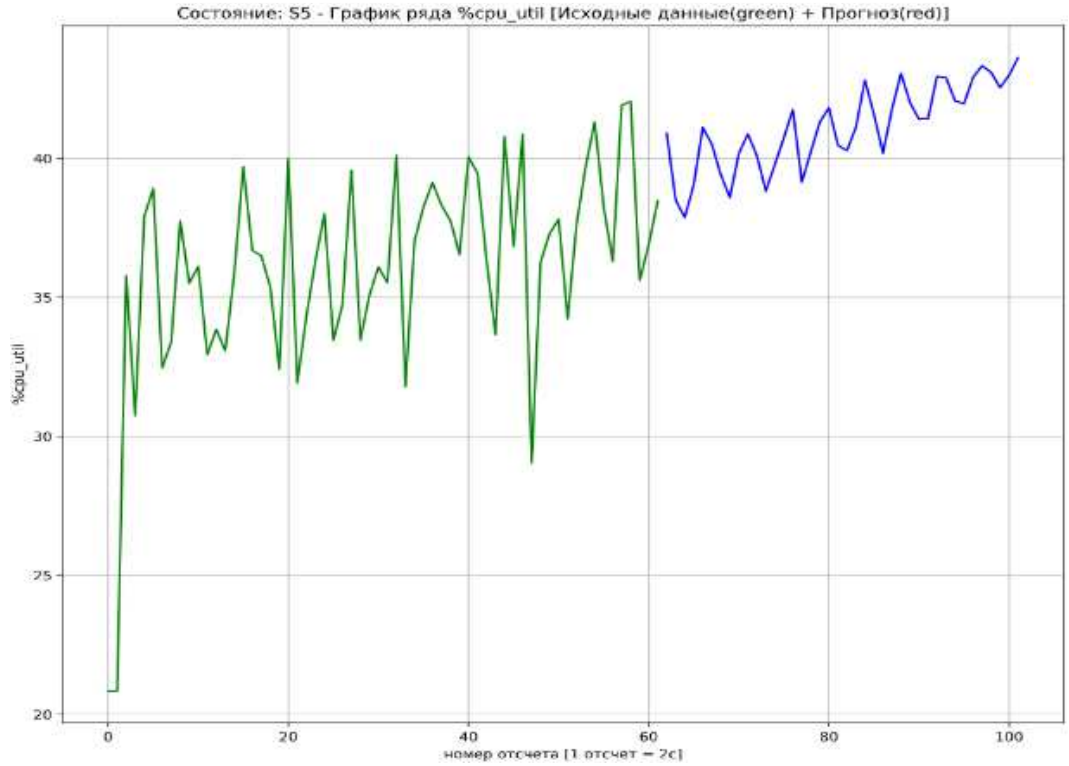

Рисунок 7. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S5

Характерным примером является ситуация, полученная для состояния S1 (рисунок 8). Мы видим реактивное изменение состояния, резкий пик около 112 отсчета (модель нейросети проиграла). Было выполнено обучение модели на данных, соответствующих первым 64 отсчетам, и был выполнен последний прогноз, но до этого уже была отправлена команда на изменение состояния системы. Эта ситуация показывает, что сложность процесса увеличивается по мере увеличения количества пользователей (числа запросов к системе), модель не учитывает изменение сложности, поскольку сложность (архитектура) самой модели не меняется.

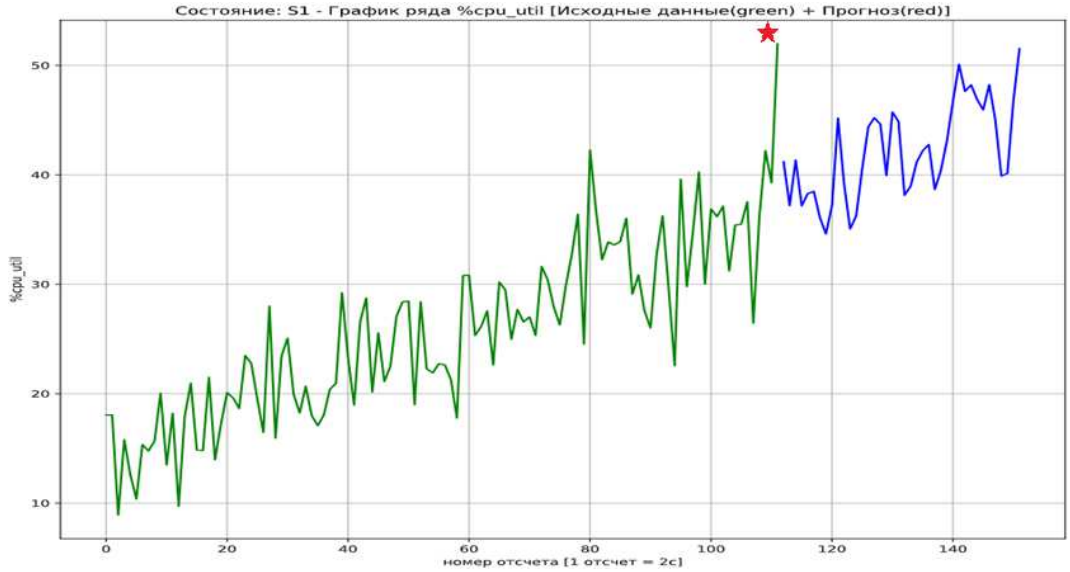

Рисунок 8. – График %CPU (зеленый – исходные данные, синий – прогноз) для состояния S1

Увеличение нагрузки (сложности сигнала) приводит к тому, что модель выполняет обобщение сигнала и теряет возможности в воспроизведении более сложной нерегулярной структуры. Модель ведет себя

подобно фильтру, который обобщает сигнал в тренд. Этот эффект хорошо заметен, если рассмотреть прогнозы моделей для состояний, когда приближаемся к максимальному количеству пользователей для системы.

Дальнейший анализ показал, что для повышения качества и точности прогнозирования временного ряда требуется создавать нейронные сети с более сложной архитектурой. Расплачиваться за усложнение архитектуры нейронной сети приходится за счет увеличения времени обучения, снижая оперативность принятия решений. Усложнение модели (увеличение количества весов) само по себе требует более ювелирного подбора гиперпараметров и может приводить к худшим результатам и переобучению.

Заключение. Результаты экспериментов показали, что предложенный подход можно использовать в практических задачах, а модель, обученная на части данных, может делать прогнозы для всей локальной области. Появляется возможность создания библиотеки нейронных моделей, способных выполнять прогнозы для различных состояний системы, выполнять дальнейшее закономерное усложнение предиктора за счет использования ансамблей моделей. Однако для точного прогнозирования сложно устроенного сигнала необходимо усложнение архитектуры предиктора. В то же время более сложная архитектура (потенциально способная к более точным предсказаниям) требует увеличения времени обучения, что снижает оперативность принятия решений. Эта ситуация наглядно показывает, что подготовка более точного прогноза для сложного сигнала требует не только улучшения алгоритма, но и более производительных вычислительных ресурсов для обеспечения оперативности принятия решений.

ЛИТЕРАТУРА

1. Straesser M., Grohmann J., von Kistowski J. et al. Why Is It Not Solved Yet?: Challenges for Production-Ready Autoscaling // In ICPE '22: 13th ACM/SPEC Intern. Conf. on Performance Engineering, Beijing, China, April 9–13, 2022. – ACM, 2022. – P. 105–115. – DOI: 10.1145/3489525.3511680.
2. Khan N., Elizondo D. A., Deka L. et al. Fuzzy Logic applied to System Monitors // IEEE Access. – 2021. – Vol. 9. – P. 56523–56538. – DOI: 10.1109/ACCESS.2021.3072239.
3. Tran D., Tran Nh., Nguyen B. M. et al. A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness // Procedia Computer Science (Intern. Conf. on Computational Science ICCS 2017). – 2017. – Vol. 108. – P. 365–374. – DOI: 10.1016/j.procs.2017.05.121.
4. Persico V., Grimaldi D., Pescape A. et al. A Fuzzy Approach Based on Heterogeneous Metrics for Scaling Out Public Clouds // IEEE Transactions on Parallel and Distributed Systems. – 2017. – Vol. 28, iss. 8. – P. 2117–2130. – DOI: 10.1109/TPDS.2017.2651810.
5. Старовойтов А. А. Моделирование систем для проактивного управления вычислительными комплексами [Электронный ресурс] // XXV Всерос. студенческая науч.-практ. конф. Нижневартовского гос. ун-та, Нижневартовск, 4-5 апр. 2023 г.) / под общ. ред. Д. А. Погонышева. – Нижневартовск: изд-во НВГУ, 2023. – Ч. 3: Информационные технологии. – С. 148–155. – URL: http://konference.nvsu.ru/konffiles/383/Stud_konf_CH3_Informacionnye_tehnologii.pdf (дата обращения 05.03.2024).
6. Старовойтов А. А. Алгоритм проактивного управления вычислительными ресурсами [Электронный ресурс] // 80-я науч. конф. студентов и аспирантов Белорус. гос. ун-та: материалы конф., Минск, 10–20 марта 2023 г.: в 3 ч. / Белорус. гос. ун-т; редкол.: А. В. Блохин (гл. ред.) [и др.]. – Минск, 2023. – Ч. 1. – 398–401 с. – URL: https://elib.bsu.by/bitstream/123456789/309836/4/80.1.pdf (дата обращения 05.03.2024).
7. Starovoytov A. A., Krasnoproshin V. V. Technology for making real-time decisions based on neural network forecasting [Electronic resource] // Pattern Recognition and Information Processing (PRIP'2023). Artificial Intelliverse: Expanding Horizons = Распознавание образов и обработка информации. Искусственная вселенная: расширяя горизонты: Proc. of the 16th Intern. Conf., Oct. 17–19, 2023, Minsk, Belarus / Belarusian State University: A. Nedzved. A. Belotserkovsky (eds.). – Minsk, 2023. – P. 58–63. – URL: https://prip.by/2023/assets/files/PRIP2023_proceedings.pdf (дата обращения 05.03.2024).

REFERENCES

1. Straesser, M., Grohmann, J., von Kistowski, J., Eismann, S., Bauer, A., & Kounev, S. (2022). Why Is It Not Solved Yet? Challenges for Production-Ready Autoscaling. In ICPE '22: *13th ACM/SPEC International Conference on Performance Engineering* (105–115). ACM. DOI: 10.1145/3489525.3511680.
2. Khan, N., Elizondo, D. A., Deka, L., & Molina-Cabello, M. A. (2021). Fuzzy Logic Applied to System Monitors. In *IEEE Access: Vol. 9* (56523–56538). IEEE. DOI: 10.1109/ACCESS.2021.3072239.
3. Tran, D., Tran, Nh., Nguyen, B. M., & Nguyen, G. (2017). A Proactive Cloud Scaling Model Based on Fuzzy Time Series and SLA Awareness. In *Procedia Computer Science (Intern. Conf. on Computational Science ICCS 2017): Vol. 108* (365–374). Elsevier. DOI: 10.1016/j.procs.2017.05.121.
4. Persico, V., Grimaldi, D., Pescapè, A., Salvi, A., & Santini, S. (2017). A Fuzzy Approach Based on Heterogeneous Metrics for Scaling Out Public Clouds. In *IEEE Transactions on Parallel and Distributed Systems: Vol. 28, iss. 8* (2117–2130). IEEE. DOI: 10.1109/TPDS.2017.2651810.
5. Starovojtov, A. A. (2023). Modelirovanie sistem dlja proaktivnogo upravlenija vychislitel'nymi kompleksami. In D. A. Pogonyshev (Ed.), *XXV Vserossijskaja studencheskaja nauchno-prakticheskaja konferencija Nizhnevartovskogo gosudarstvennogo universiteta: Ch. 3. Informatsionnye tekhnologii* (148–155). Nizhnevartovsk: Publ. NVGU. http://konference.nvsu.ru/konffiles/383/Stud_konf_CH3_Informacionnye_tehnologii.pdf. (In Russ.).

6. Starovojtov, A. A. (2023). Algoritm proaktivnogo upravlenija vychislitel'nymi resursami. In A. V. Blohin (Ed.) et al. 80-ja nauchnaja konferencija studentov i aspirantov Belorusskogo gosudarstvennogo universiteta: Ch. 1 (398–401). Minsk: Publ. BGU. http://konference.nvsu.ru/konffiles/383/Stud_konf_CH3_Informacionnye_tehnologii.pdf. (In Russ.).
7. Starovoytov, A. A., & Krasnoproshin, V. V. (2023). Technology for making real-time decisions based on neural network forecasting. In A. Nedzved, & A. Belotserkovsky (Eds.), *Pattern Recognition and Information Processing (PRIP'2023). Artificial Intelliverse: Expanding Horizons: Proceedings of the 16th International Conference* (58–63). Minsk: Publ. BSU. https://prip.by/2023/assets/files/PRIP2023_proceedings.pdf.

Поступила 14.03.2024

REAL-TIME MANAGEMENT OF CRITICAL IT-SYSTEMS BASED ON NEURAL NETWORK TECHNOLOGIES

A. STAROVOYTOV, V. KRASNOPROSHIN
(Belarusian State University, Minsk)

The paper investigates a relevant applied problem associated with building decision support systems for critical information services. An original approach is proposed, based on neural network forecasting, within which a method of dynamic local approximation using neural network models has been developed. The principles of constructing and implementing the operational algorithm (under conditions of uncertainty of the external load profile) of a combined proactive system for managing computational resources are outlined. Experiments have been conducted that confirm the effectiveness of the method and the approach as a whole.

Keywords: *operational decision-making, critical information system, proactive management, uncertainty of external load, neural networks, neuromodule.*