УДК 519.872.7
DOI: 10.26102/2310-6018/2026.56.5.007

Прогнозирование нагрузки на микросервисную систему с использованием метода ARIMA и байесовских сетей

В.Р. Четвертухин

Белгородский государственный технологический университет им. В.Г. Шухова, Белгород, Российская Федерация

Резюме. В статье представлен подход прогнозирования нагрузки на микросервисную систему, объединяющий методы анализа временных рядов ARIMA и вероятностный вывод в байесовских сетях. Он позволяет учитывать как шаблоны нагрузки на отдельные микросервисы во времени, так и структурные зависимости между этими микросервисами. Представленный подход состоит из двух этапов: на первом этапе ARIMA-модели строят независимые прогнозы для каждого микросервиса, а на втором этапе байесовская сеть корректирует полученные прогнозы с учетом зависимостей между микросервисами и распространения нагрузки от сервиса к сервису. Итоговый прогноз состоит из взвешенных результатов обоих этапов. Кроме того, предусмотрен критерий обнаружения аномальной нагрузки, благодаря которому система прогнозирования способна реагировать на нее, изменяя веса и другие параметры алгоритма. Проводится экспериментальная проверка программной реализации подхода на данных реальных систем. Результаты сравниваются, в первую очередь, с изолированным применением ARIMA и с применением Long-Short Term Memory (LSTM) сетей для той же задачи. Алгоритм показывает себя перспективным к применению в задаче прогнозирования нагрузки.

Ключевые слова: прогнозирование нагрузки, временные ряды, ARIMA, байесовские сети, микросервисная архитектура.

Для цитирования: Четвертухин В.Р. Прогнозирование нагрузки на микросервисную систему с использованием метода ARIMA и байесовских сетей. *Моделирование, оптимизация и информационные технологии.* 2026;14(5). URL: https://moitvivt.ru/ru/journal/article?id=2284 DOI: 10.26102/2310-6018/2026.56.5.007

Forecasting the load on a microservice system using the ARIMA method and Bayesian networks

V.R. Chetvertukhin

Belgorod State Technological University named after V.G. Shukhov, Belgorod, the Russian Federation

Abstract. The article presents an approach to predicting the load on a microservice system that combines ARIMA time series analysis methods and probabilistic inference in Bayesian networks. This approach allows for the consideration of both the load patterns on individual microservices over time and the structural dependencies between these microservices. The presented approach consists of two stages: in the first stage, ARIMA models build independent forecasts for each microservice, and in the second stage, a Bayesian network adjusts the obtained forecasts, taking into account dependencies between microservices and the propagation of load from service to service. The final forecast consists of the weighted results of both stages. In addition, an anomaly detection criterion is provided, which allows the forecasting system to respond to anomalies by changing the weights and other parameters of the algorithm. The approach is experimentally tested using real-world system data. The results are compared with the isolated use of ARIMA and the use of Long-Short Term Memory (LSTM) networks for the same task. The algorithm shows promise for use in load forecasting.

Keywords: load forecasting, time series, ARIMA, Bayesian networks, microservice architecture.

For citaition: Chetvertukhin V.R. Forecasting the load on a microservice system using the ARIMA method and Bayesian networks. *Modeling, Optimization and Information Technology.* 2026;14(5). (In Russ.). URL: https://moitvivt.ru/ru/journal/article?id=2284 DOI: 10.26102/2310-6018/2026.56.5.007

Введение

Эффективность работы микросервисной системы критически зависит от нагрузки на эту систему во времени. Поэтому наличие модуля прогнозирования нагрузки, влияющего на масштабирование ресурсов по надобности, в современных микросервисных системах является необходимым. Обеспечение точности прогноза в таких модулях – не самая простая задача: в отличие от монолитных приложений, в которых достаточно прогнозировать общий входящий трафик, архитектура, построенная на микросервисах, требует предсказания нагрузки на каждый из десятков, а иногда даже сотен взаимодействующих друг с другом сервисов [1].

Задача прогнозирования нагрузки на систему не является новой, поэтому сформировались традиционные подходы к ее решению. Эти подходы можно условно разделить на три категории. К первой из них можно отнести статистические методы, основанные на временных рядах, среди которых можно выделить ARIMA [2], экспоненциальное сглаживание [3] и сезонную декомпозицию [4]. Такие методы настроены на временные шаблоны, но имеют важный недостаток – структурные связи между компонентами системы, нагрузка на которую прогнозируется, не учитываются.

Ко второй категории относятся методы машинного обучения, такие как нейронные сети LSTM [5], случайные леса [6] и градиентный бустинг [7]. Эти популярные в последнее время технологии решают недостаток первой категории методов: учесть в прогнозе можно любые зависимости и условия, влияющие на систему [8]. Однако, помимо того, что методам машинного обучения необходимы большие объемы данных о структуре системы и ее поведении в прошлом, всегда стоит учитывать, что понять в точности, на основе чего был дан тот или иной прогноз, будет невозможно, так как нейронные сети работают по принципу «черного ящика». Соответственно, принимаемые решения на основе таких прогнозов можно назвать непредсказуемыми и неинтерпретируемыми, поэтому выбор в пользу методов машинного обучения может быть опасным для управления критически важными системами.

Третья категория – решения, основанные на вероятностных графических моделях (ВГМ), к которым можно отнести скрытые марковские модели [9] или байесовские сети [10]. Решения, принимаемые системами, построенными на основе таких моделей, уже можно проследить. Они так же, как и нейронные сети, позволяют учесть связи между компонентами системы. Но учет временной динамики нагрузки для таких моделей не является традиционной задачей.

Таким образом, можно выделить ряд важных условий, учет влияния которых в одной единой системе прогнозирования нагрузки на микросервисную систему позволит получать наиболее точный и интерпретируемый результат:

1. Отказ от изолированного рассмотрения микросервисов. Независимый прогноз по каждому компоненту системы может дать далекий от истины результат, поскольку в реальности нагрузка на один из микросервисов может повлиять на нагрузку других, зависимых от него сервисов с некоторой задержкой и коэффициентом усиления. Например, если рассмотреть распространенную систему интернет-магазина, в которой можно выделить сервисы каталога товаров, корзины и оплаты, становится очевидным, что всплеск запросов к сервису каталога повлечет за собой каскадный всплеск к следующим двум сервисам.

2. Интерпретируемость принимаемых решений. Как уже было отмечено, принятие решений по прогнозу, который был получен неизвестным путем, может подойти не для всех систем.

3. Случайность и динамика во времени. В реальных системах отсутствие учета этих параметров делают прогноз бесполезным.

Для реализации модели прогнозирования, учитывающей эти условия, в работе предлагается гибридный подход, который объединяет в себе сильные стороны временного анализа нагрузки (ARIMA) и моделирования структуры микросервисной системы со структурными связями между сервисами и учетом случайности (байесовские сети). Для описания этого подхода в статье были выделены следующие задачи:

1) формализация задачи прогнозирования в контексте микросервисной системы с учетом зависимостей;

2) описание алгоритма ARIMA-BN: этапы и обнаружение аномалий;

3) приведение наиболее важных участков программной реализации;

4) экспериментальная валидация алгоритма.

Материалы и методы

Опишем формальную модель системы. Обозначим микросервисную систему как набор сервисов $S = \{s_1, \ldots, s_n\}$ с графом зависимостей $G = (S, E)$, где ребра E представляют собой вызовы между микросервисами.

Для каждого сервиса s_i определим временной ряд нагрузки L_i:

$$L_i = \{l_i(t): t = 1,2,\ldots,T\},$$

где $l_i(t)$ – количество запросов к сервису i в момент времени t.

По формальной модели поставим следующую задачу: по историческим данным $\{L_i\}_{t=1}^{T}$ спрогнозировать будущие значения $\{\widehat{L_i}\}_{t=T+1}^{T+h}$, где h – горизонт прогноза.

Теперь опишем межсервисные зависимости в контексте нагрузки на систему. Нагрузка распространяется по графу зависимостей G с задержками и коэффициентами:

$$l_j(t) = l_j^{internal}(t) + \sum_{i:(i,j)\in E} \alpha_{ij} \cdot l_i\big(t - \tau_{ij}\big) + \varepsilon_j(t),$$

где $l_i^{internal}(t)$ – собственная нагрузка сервиса (прямые обращения); α_{ij} – коэффициент передачи нагрузки от i к j; τ_{ij} – задержка распространения; $\varepsilon_j(t)$ – случайный шум.

Поскольку работу программной реализации планируется сравнивать с работой современных нейронных моделей (LSTM), для оценки качества прогноза будут использованы принятые в машинном обучении метрики. Первая из них – среднеквадратичная ошибка Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n \cdot h}\sum_{i=1}^{n}\sum_{t=T+1}^{T+h}\Big(l_i(t) - \widehat{l_i}(t)\Big)^2}.$$

Также для оценки будет использована средняя абсолютная процентная ошибка Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{100\%}{n \cdot h}\sum_{i=1}^{n}\sum_{t=T+1}^{T+h}\left|\frac{l_i(t) - \widehat{l_i}(t)}{l_i(t)}\right|.$$

И, наконец, коэффициент детерминации, который покажет объяснимую долю общей дисперсии:

$$R^2 = 1 - \frac{\sum_{i,t}\Big(l_i(t) - \widehat{l_i}(t)\Big)^2}{\sum_{i,t}(l_i(t) - \overline{l_i})^2}.$$

Поскольку суть работы алгоритма прогнозирования состоит в двух этапах работы (ARIMA-прогнозирование для каждого микросервиса и байесовская коррекция), далее в статье он будет называться ARIMA-BN (Bayesian Network). Реализация предлагаемого алгоритма разбита на четыре основных компонента:

1) модуль ARIMA – независимое прогнозирование для каждого сервиса;

2) реализация байесовской сети, отражающей влияние прогнозов нагрузки одних микросервисов на нагрузку других;

3) модуль коррекции, объединяющий результат работы первого и второго модуля;

4) детектор аномалий, выявляющий нестандартные шаблоны поведения.

Ниже будет представлено описание каждого компонента.

ARIMA. Для каждого микросервиса строится модель ARIMA (p, d, q):

$$\phi(B)(1-B)^d l_i(t) = \theta(B)\varepsilon_i(t).$$

Здесь B – оператор сдвига назад:

$$B\big(l_i(t)\big) = l_i(t-1),$$

$$B^k\big(l_i(t)\big) = l_i(t-k),$$

ϕ – полином авторегрессии (AR):

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p.$$

θ – полином скользящего среднего (MA), ε – белый шум: последовательность некоррелированных случайных величин с нулевым средним и постоянной дисперсией. Раскрытие дает:

$$\theta(B)\varepsilon_i(t) = \varepsilon_i(t) + \theta_1\varepsilon_i(t-1) + \ldots + \theta_q\varepsilon_i(t-q).$$

Байесовская сеть. Узел байесовской сети представляет собой прогноз нагрузки в момент $t + k$ (т. е. через k шагов от текущего момента t):

$$\left\{L_i^{t+k}\right\}_{i=1,k=1}^{n,h}.$$

Условные вероятности узлов распределены нормально:

$$P\left(L_j^{t+k} \mid Parents\big(L_j^{t+k}\big)\right) = \mathcal{N}\big(\mu_{j,k}, \sigma_{j,k}^2\big).$$

Здесь *Parents* – множество узлов, от которого рассматриваемый узел зависит. Выбор в пользу нормального распределения был сделан потому, что количество запросов в единицу времени хорошо аппроксимируется гауссианой по центральной предельной теореме.

Математическое ожидание распределения определяется так:

$$\mu_{j,k} = \hat{l}_j^{ARIMA}(t+k) + \textstyle\sum_{i:(i,j)\in E} \beta_{ij} \cdot \left(L_i^{t+k-\tau_{ij}} - \hat{l}_i^{ARIMA}\big(t+k-\tau_{ij}\big)\right).$$

Первое слагаемое – это «базовый» прогноз, полученный от модуля ARIMA. Байесовская сеть не строит прогноз с нуля, а принимает результат ARIMA как отправную точку.

Второе слагаемое – это сумма коррекций от родительских сервисов. Для каждого сервиса s_i, от которого зависит сервис s_j, вычисляется выражение:

$$L_i^{t+k-\tau_{ij}} - \hat{l}_i^{ARIMA}\big(t+k-\tau_{ij}\big).$$

Это разность между фактическим (или уже скорректированным байесовской сетью) значением нагрузки на s_i и тем, что предсказывала ARIMA. Иными словами, это отклонение от ARIMA-прогноза – «новая информация», которую ARIMA не уловила.

Коэффициент β отражает влияние отклонения нагрузки родительского сервиса от прогноза ARIMA на прогноз нагрузки на рассматриваемый сервис. Этот коэффициент оценивается из исторических данных.

Объединение прогнозов. Окончательный прогноз нагрузки основывается на сумме прогнозов ARIMA и BN, каждый из которых имеет вес:

$$\hat{l}_i^{final}(t+k) = \omega_i^{ARIMA} \cdot \hat{l}_i^{ARIMA}(t+k) + \omega_i^{BN} \cdot \mathbb{E}\left[L_i^{t+k} \mid evidence\right].$$

Здесь первое слагаемое – взвешенный вклад чистого ARIMA-прогноза. Второе слагаемое – математическое ожидание нагрузки по апостериорному распределению байесовской сети, вычисленное с учетом всех доступных свидетельств (evidence), умноженное на вес. Свидетельства включают ARIMA-прогнозы для всех сервисов, а также наблюдаемые данные о текущих отклонениях. Вероятностный вывод в байесовской сети распространяет эту информацию по графу зависимостей и выдает скорректированное математическое ожидание.

Сумма весов равна 1, при этом какой из двух прогнозов получит больший вес, определяется формулой адаптивных весов:

$$\omega_i^{ARIMA} = \frac{1/MSE_i^{ARIMA}}{1/MSE_i^{ARIMA} + 1/MSE_i^{BN}}.$$

Эта формула реализует принцип обратной пропорциональности ошибке: компоненту с меньшей ошибкой присваивается больший вес.

Обнаружение и обработка аномалий. Аномалия фиксируется в случае выполнения одного из двух критериев:

$$Anomaly(t) = \begin{cases} \text{есть, если } \left|l_i(t) - \widehat{l_i}(t-1)\right| > 3\sigma_i, \text{ или } \chi^2 > \chi^2_{critical} \\ \text{нет} \end{cases}.$$

Первый критерий – «правило трех сигм»: если ошибка прогноза превысила три стандартных отклонения, это событие крайне маловероятно при нормальном режиме работы и квалифицируется его как аномалия. Этот критерий отлавливает точечные выбросы – внезапные резкие скачки нагрузки.

Второй критерий – статистика хи-квадрат, проверяющая независимость ошибок прогноза. Этот критерий отлавливает не точечные выбросы, а системные изменения, например, постепенное изменение характера нагрузки, которое модель ARIMA не может учесть с текущими параметрами.

При обнаружении аномалии алгоритм выполняет следующие действия:

1) увеличивает вес байесовской компоненты, поскольку в аномальной ситуации критически важным становятся межсервисные зависимости для понимания каскадных эффектов;

2) уменьшает горизонт прогнозирования, поскольку в условиях аномалии долгосрочный прогноз ненадежен;

3) переобучает ARIMA и перерассчитывает коэффициенты β на более коротком окне данных.

Программная реализация. Ниже представлен верхнеуровневый программный код основного алгоритма ARIMA-BN на языке Python:

```
def hybrid_forecast(data, graph, horizon):
  forecasts = {}
  # Фаза 1: ARIMA для каждого сервиса
```

```
    for service in graph.nodes():
        series = data[service]
        params = select_arima_parameters(series)
        model = ARIMA(series, order=params)
        fitted = model.fit()
        forecasts[service] = fitted.forecast(steps=horizon)
    # Фаза 2: Построение байесовской сети
    bn = build_dynamic_bayesian_network(graph, horizon)
    # Оценка параметров зависимостей
    for (i, j) in graph.edges():
        delay = estimate_delay(data[i], data[j])
        coefficient = estimate_coefficient(data[i], data[j], delay)
        bn.set_edge_params(i, j, delay, coefficient)
    # Фаза 3: Вероятностный вывод
    evidence = prepare_evidence(forecasts)
    posterior = bn.inference(evidence)
    # Фаза 4: Объединение прогнозов
    final_forecast = {}
    for service in graph.nodes():
        arima_weight = calculate_weight(service, 'arima')
        bn_weight = calculate_weight(service, 'bn')
        final_forecast[service] = (
            arima_weight * forecasts[service] +
            bn_weight * posterior[service]
        )
    return final_forecast
```

Функция оценки задержки из этого алгоритма представлена в следующем листинге:

```
def estimate_delay(series_i, series_j, max_lag=10):
    correlations = []
    for lag in range(max_lag + 1):
        if lag == 0:
            corr = np.corrcoef(series_i, series_j)[0, 1]
        else:
            corr = np.corrcoef(series_i[:-lag], series_j[lag:])[0, 1]
        correlations.append(corr)

    return np.argmax(correlations)

def estimate_coefficient(series_i, series_j, delay):
    if delay == 0:
        X = series_i.reshape(-1, 1)
        y = series_j
    else:
        X = series_i[:-delay].reshape(-1, 1)
        y = series_j[delay:]

    model = LinearRegression()
    model.fit(X, y)
    return model.coef_[0]
```

Временная сложность гибридного алгоритма составляет:

$$O(n \cdot T \log T + n^2 \cdot h^2),$$

где n – число микросервисов, T – длина истории, h – горизонт прогноза.

Оцененная сложность представляет собой сумму сложностей двух фаз: первое слагаемое – фаза ARIMA, второе слагаемое – фаза байесовской сети.

Результаты

Эксперимент проводился на наборе данных собственной микросервисной системы онлайн-заказов товаров. Период данных – 30 дней (январь 2026), количество микросервисов – 15, частота дискретизации – 1 минута, объем записей – 648 тысяч.

Граф зависимостей микросервисов системы с разбиением по уровням представлен на Рисунке 1.

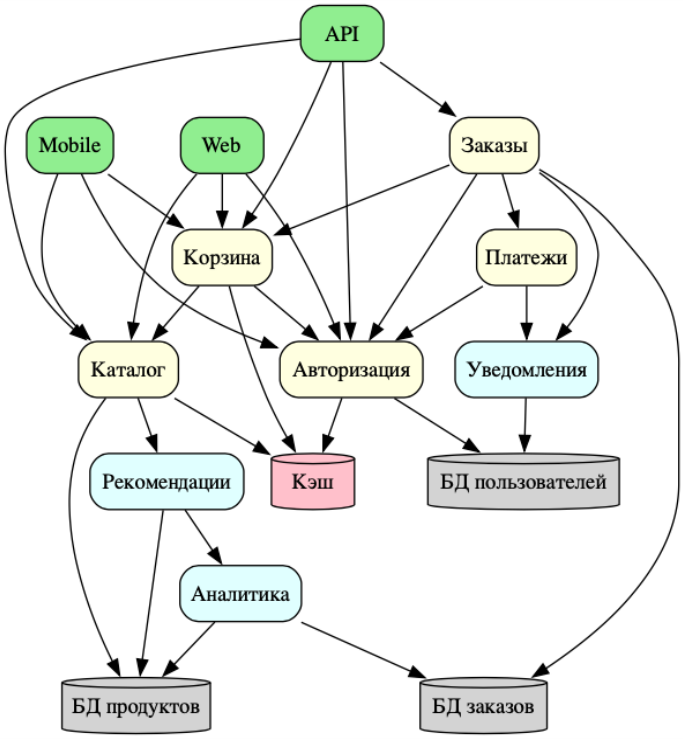

Рисунок 1 – Граф зависимостей микросервисной системы онлайн-заказов товаров
Figure 1 – Dependency graph of a microservice system for online product orders

Алгоритм сравнивался с классической моделью ARIMA без учета зависимостей и рекурентной нейронной сетью с долгой краткосрочной памятью (LSTM).

Результаты сравнения работы алгоритмов по вышеупомянутым критериям (RMSE, MAPE, R2) для набора данных представлены в Таблице 1.

Таблица 1 – Сравнение методов на данных набора
Table 1 – Comparison of methods on the data set

Метод	RMSE	MAPE, %	R^2	Время (сек)
ARIMA	187,3	12,4	0,71	2,1
LSTM	162,5	10,8	0,78	45,3
ARIMA-BN	128,9	8,5	0,86	12,3

Обсуждение

Эксперимент подтвердил, что комбинирование ARIMA и байесовских сетей имеет смысл: точность прогноза по RMSE улучшилась более чем на 31 % по сравнению с классической моделью ARIMA. Кроме того, прогноз оказался точнее, чем у сети LSTM, и работа алгоритма по времени оказалась существенно быстрее.

Заключение

В работе предложен гибридный алгоритм прогнозирования нагрузки ARIMA-BN, объединяющий временной анализ с моделированием структурных зависимостей. Ключевые достижения:

1) формализована задача прогнозирования нагрузки с учетом межсервисных зависимостей;

2) разработана программная реализация предложенного подхода;

3) реализован механизм обработки аномалий;

4) экспериментально подтверждено снижение ошибки прогноза на 31 % по сравнению с ARIMA.

Результаты работы дают почву для проведения экспериментов на больших наборах данных крупных микросервисных систем. Кроме того, планируется интеграция разработанной программы с системами оркестрации.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Груничев Ю.А., Коняшкин Р.А. Облачные вычисления и их влияние на архитектуру современных информационных систем. *Парадигма*. 2026;(1-1):85–89.
2. Box G.E.P., Jenkins G.M., Reinsel G.C., Ljung G.M. *Time Series Analysis: Forecasting and Control*. Hoboken: John Wiley & Sons; 2015. 720 p.
3. Сахаров Д.В., Гельфанд А.М., Казанцев А.А., Пестов И.Е. Использование математических методов прогнозирования для оценки нагрузки на вычислительную мощность IoT-сети. *Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России.* 2020;(2):86–94.
   Saharov D.V., Gelfand A.M., Kazantsev A.A., Pestov I.E. Using mathematical forecasting methods to estimate the load on the computing power of the IoT network. *Vestnik Saint-Petersburg University of State Fire Service of Emercom of Russia.* 2020;(2):86–94. (In Russ.).
4. Cleveland R.B., Cleveland W.S., McRae J.E., Terpenning I. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics.* 1990;6(1):3–73.
5. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997;9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
6. Breiman L. Random forests. *Machine Learning*. 2001;45(1):5–32. https://doi.org/10.1023/A:1010933404324
7. Friedman J.H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*. 2001;29(5):1189–1232. https://doi.org/10.1214/aos/1013203451
8. Терехин М.А., Иващенко А.В., Кулаков Г.А. Концептуальный подход к интеграции искусственного интеллекта в инженерную деятельность. *Моделирование, оптимизация и информационные технологии*. 2025;13(2). https://doi.org/10.26102/2310-6018/2025.49.2.031
   Terekhin M.A., Ivaschenko A.V., Kulakov G.A. A conceptual approach to the integration of artificial intelligence into engineering activities. *Modeling, Optimization and*

Information Technology. 2025;13(2). (In Russ.). https://doi.org/10.26102/2310-6018/2025.49.2.031
9. Rabiner L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*. 1989;77(2):257–286. https://doi.org/10.1109/5.18626
10. Koller D., Friedman N. *Probabilistic Graphical Models: Principles and Techniques.* Cambridge: MIT Press; 2009. 1270 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Четвертухин Виктор Романович, ассистент Белгородский государственный технологический университет им. В.Г. Шухова, Белгород, Российская Федерация.
e-mail: victor.chet@mail.ru
ORCID: 0009-0003-6110-1227

Victor R. Chetvertukhin, Assistant, Belgorod State Technological University named after V.G. Shukhov, Belgorod, the Russian Federation.

Статья поступила в редакцию 18.03.2026; одобрена после рецензирования 07.05.2026; принята к публикации 15.05.2026.

The article was submitted 18.03.2026; approved after reviewing 07.05.2026; accepted for publication 15.05.2026.