ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.896

О.О. ДРУЖИНІНА, Р.Н. КВЄТНИЙ

Вінницький національний технічний університет, Вінниця

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ ВЕБ-СЕРВЕРІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ НА ОСНОВІ НЕЙРОМЕРЕЖ

Анотація. Здійснено огляд та аналіз способів підвищення якості обслуговування веб-серверів. Запропоновано підхід інтелектуального кешування даних. Запропоновно підхід до підвищення ефективності функціонування веб-серверів з використанням технології прогнозування часових рядів на основі нейронних мереж радіально-базисного типу.
Ключові слова: якість обслуговування веб-серверів, прогнозування, нейронні мережі, GRNN, PNN, часові ряди, кешування.
Аннотация. Приведен обзор и анализ существующих способов повышения качества обслуживания веб-серверов. Предложен подход интеллектуального кеширования данных. Предложен подход к повышению эффективности функционирования веб-серверов с использованием технологии прогнозирования временных рядов на основе нейронных сетей радиально-базисного типа.
Ключевые слова: качество обслуживания веб-серверов, погнозирование, нейронные сети, GRNN, PNN, временные ряды, кеширование.
Abstract. Web-servers quality of service approaches overviewed. Intelectual data caching approach was proposed. Approach for web-servers efficiency improving, based on time series forecasting with radial-basis neural networks was proposed.
Key words: web-servers Quality of Service, forecasting, neural networks, GRNN, PNN, time series, caching.

Вступ

На ефективність функціонування веб-додатків значним чином впливає якість обслуговування (Quality of Service, QoS) веб-серверів.

Веб-сервер – це апаратно-програмна платформа, підключена до локальної або глобальної мережі, на якій буде розміщено програмний модуль або групу модулів, які здійснюватимуть обробку клієнтських запитів, а також формуватимуть відповіді на ці запити у відповідності до логіки, що реалізована у цих модулях. Апаратну частину веб-сервера становить серверний комп’ютер, програмну – операційна система та набір службових додатків, а також набір модулів, що реалізують логіку обробки клієнтських запитів. Службові модулі – це модулі, що реалізують роботу з мережевими інтерфейсами і надають необхідне програмне оточення для роботи модулів обробки клієнтських запитів.

Серед найбільш розповсюджених службових додатків веб-серверів можна виділити наступні: Apache, Nginx, Microsoft IIS. Зазвичай, веб-сервери отримують запити через мережу по протоколу HTTP (або HTTPS у випадку використання захищеного каналу передачі) або через інші протоколи, надбудовані над ним, наприклад XML-RPC чи SOAP, тощо.Також веб-сервери можуть використовувати базу даних чи інші репозиторії для зберігання необхідних даних. Зазвичай, база даних розташована на окремому сервері, і обмін даними з веб-сервером здійснюється через локальну чи глобальну мережу за допомогою протоколів, специфічних для систем управління базами даних. Отже, веб-сервер є достатньо складним об’єктом, коректна робота та швидкодія якого залежить від багатьох чинників.

До головних метрик якості обслуговування веб-серверів відносяться: час відгуку та собівартість[1]. Зростання Інтернет обчисленнь призвело до зростання обчислювальних центрів, внаслідок чого корпоративні центри обробки даних, як правило, складаються з великої кількості серверів, які не використовуються в повній мірі. Це в свою чергу стало причиною збільшення вартості обслуговання відповідних сервісів, що обумовлено прямою залежністю між кількістю серверів та витратами на їх аренду, електроенергію, ліценції на програмне забезпечення та адміністрування.Саме тому насьогодні зростає кількість корпорацій, які використовують пули серверних ресурсів, що стало можливим завдяки розвитку віртуалізації серверів, мереж та пристроїв збереження інформації. Сереверний пул являє собою групу серверів в кластері, які об’єднані в деяке логічне одиницю(пул). Ідея серверних пулів полягає у відсутності прив’язки ресурсів в кластері до конкретних фізичних серверів. Негативною особливостю такої системи є складність її адміністрування, оскільки для ефективного використання серверних пулів необхідно здійснювати аналіз навантажень, прогнозування навантаження на окремий ресурс та ефективний перерозподіл ресурсів.

Задачу ефективного управління пулами ресурсів можна розділити на декілька простіших підзадач:

1) управління доступом до ресурсів з метою визначення чи є в пулі достатньо ресурсів для додання нового навантаження;

2) розміщення робочого навантаження на програмні додатки з метою отримання рекомендації щодо оптимального розміщення навантаження для зменшення кількості використовємих серверів.

3) прогнозування потреб навантаження.

В даній роботі увага зосердиться на останній задачі прогнозування потреб навантаження. Аналіз розглянутих літературних джерел з даної проблематики показав, що наразі існує невідповідність існуючих інформаційних технологій прогнозування навантаження веб-серверів сучасним вимогам щодо спів-

відношення вартість/ефективність, що спричинено використанням недостатньо точних методів та моделей прогнозу, які базуються на спрощеному математичному апараті [2, 3]. Саме тому є доцільним розробити технологію прогнозування серверного навантаження, яка б базувалась на підходах та методах ідентифікації складних нелінійних систем та дослідити її ефективність ефективність у порівнянні з висвітленими у літературних джерелах.

За час відгуку веб-сервера відносно клієнтської сторони приймемо час, необхідний для відправки запиту на сервер та одержання і обробку відповіді клієнтом. Часом, що знадобиться на те, щоб попередньо підготувати до відправки клієнтський запит, а також часом, витраченим на обробку відповіді, можна знехтувати, оскільки дані часові характеристики залежать виключно від клієнтської сторони, і жодним чином не стосуються веб-серевера. Час відгуку веб-сервера може залежати від декількох факторів, кожен з яких може впливати істотним чином не величину часового інтервалу між запитом до сервера і відповіддю сервера. До таких факторів можна віднести:

- якість мережевого підключення, від якої залежить безпосередня швидкість передач даних;
- швидкодія апаратної платформи, на якій розташований веб-сервер;
- характеристики операційної системи на якій розгорнуто веб-сервер;
- архітектура та реалізація службового додатку веб-сервера (швидкодія, ефективність, оптимальність та потреба в системних ресурсах, тощо);
- швидкодія модулів, за допомогою яких веб-сервер здійснює взаємодію із зовнішніми ресурсами, необхідними для його коректного функціонування;
- архітектура та реалізація системи управління базою даних, яка застосовується для зберігання даних, необхідних веб-серверу;
- архітектура та реалізація модулів обробки клієнтських запитів (рис 1).

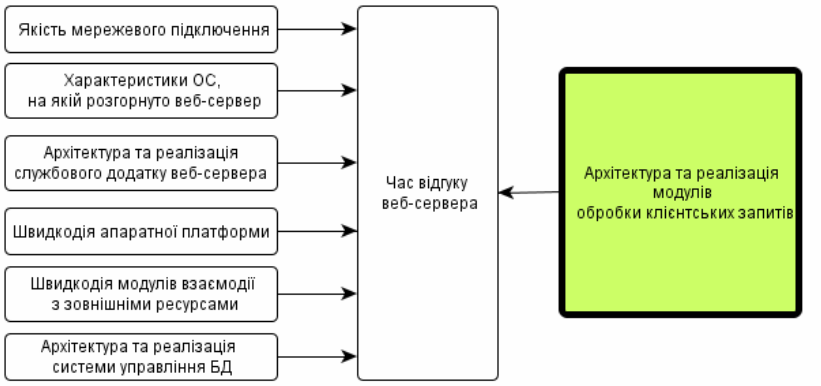


Рисунок 1– Фактори, які впливають на час відгуку веб-сервера

В даній роботі ми не будемо зосереджувати увагу на апаратних і мережевих факторах, а також факторах, пов’язаних із операційною системою, службовим додатком чи системою управління базо даних. Зосередимось на модулі обробки клієнтських запитів.

Модуль обробки клієнтських запитів містить реалізацію роботи сервісу, який вирішує ті чи задачі або надає клієнтам необхідні дані. Як показує практика, при правильному проектуванні і ефективній реалізації сервісного модуля, «вузьким місцем» в ньому стає комунікація з зовнішніми компонентами чи серверами, наприклад, базою даних. Тобто, основна частка часу, що витрачається на обробку клієнтського запиту, йде на встановлення зв’язку між веб-сервером і зовнішніми серверами чи компонентами, а також на передачу даних між ними.

Якщо в якості зовнішнього компонента розглядати базу даних, то значна частина часу витрачається на встановлення з’єднання з базою даних та обробку запитів – їх розбір, компіляцію, визначення і обрання оптимального плану виконання, очікування зняття блокувань при доступі до ізольованих даних, вибірку даних, тощо.

Для вирішення проблеми тривалого очікування з’єднання з базою даних застосовують підхід при якому формують «пули» з’єднань з базою даних, які використовуються повторно. При необхідності з’єднання з базою даних, компонент, який забезпечує це з’єднання, звертається до спеціального контейнера, в якому містяться наявні активні з’єднання з базою даних. Якщо цей контейнер порожній, компонент створює нове з’єднання, повертає його компоненту, який його потребує, і після того я к це

з’єднання було використано, не закриває його, а переміщує його у вищевказаний контейнер. Таким чином одне з’єднання з базою даних може бути використано багаторазово без накладних витрат, пов’язаних із його створенням та закриттям.

Для вирішення проблеми тривалої обробки запитів на стороні бази даних використовують технологію кешування даних, які не змінюються на протязі певного інтервалу часу. Наприклад, якщо два запити до бази даних з однаковими умовами повертають однаковий результат, то доцільним є розміщення даного результат у кеші, і виключення звернення до бази даних, що значно прискорює роботу веб-сервера.

Розглянемо кешування більш докладно. Кеш являє собою окремий фізичний сервер, на якому виконується лише додаток, що забезпечує кешування даних. Кешування полягає у розміщенні об’єкта, що буде закешований в оперативній пам’яті кеш-сервера. Закешований об’єкт маркується відповідним ключовим значенням (яке обирає клієнт кеша) і з використанням якого в подальшому відбувається доступ до закешованого об’єкта. Оскільки робота і пошук в оперативній пам’яті є значно швидшими ніж дискові операції – кеш значно прискорює отримання даних, які не змінюються протягом певного часу. Розглянемо роботу кешування на прикладі програмного додатку *Memcached*, який насьогодні є найбільш розповсюдженим і еталонним рішенням для кешування даних [4].

Memcached є кросплатформеним додатком, який запускається на цільовому сервері та взаємодіє з зовнішніми клієнтами за допомогою спеціального протоколу. Клієнти, які бажають працювати з додатком *Memcached* повинні містити відповідні клієнти, які можуть взаємодіяти з додатком *Memcached*, наприклад *xmemcached*. Клієнт передає *Memcached* команду *set* і таким чином здійснює розміщення необхідного об’єкту в кеші. Параметрами цього методу виступають ключ, за яким відбуватиметься пошук об’єкта в кеші, сам об’єкт, який необхідно закешувати, а також дата, коли об’єкт «застаріє» і його значення необхідно буде актуалізувати на основі даних з бази даних або інших джерел. Закешован об’єкти в процесі роботи можуть займати весь обсяг оперативної пам’яті, яка була виділена додатком *memcached* при запуску. Коли об’єкт стає «застарілим», він продовжує зберігатись в пам’яті кешу до тих пір, доки не постане необхідність для «кешування» нових об’єктів, або не відбудеться звернення до даного об’єкту і він буде видалений.

Виходячи з вищенаведеного, ми бачимо, що існуюча концепція кешування має ряд недоліків, зокрема:

- захаращення кешу об’єктами, які ще не «застаріли», проте ніким не використовуються, і як наслідок неможливість закешувати нові об’єкти;
- використання всієї доступної оперативної пам’яті додатком *Memcached* в той час, коли для активних закешованих об’єктів необхідний менший обсяг оперативної пам’яті, неможливість використати цю пам’ять для інших цілей;
- заповнення контейнеру ключів закешованих об’єктів «зайвими» ключами об’єктів, до яких не буде звернення, і відповідно зі зростанням розміру контейнера, збільшенням часу пошуку необхідного ключа;
- чутливість до помилок клієнтів в налаштуванні тривалості кешування об’єктів – при обранні невмотивовано великого строку кешування, захаращення кешу непотрібними об’єктами.

Отже, як видно з вищенаведених недоліків, основним «вузьким місцем» кешу є розмір дозволеної до використання оперативної пам’яті. Дану проблему було вирішено через «приєднання» до кешу, розміщеного в оперативній пам’яті, простору на жорсткому диску. Дане було реалізовано у 2010 році у проекті *Membase*, який в наш час перейменовано в *Couchbase*. Додаток *Couchbase* є повністю сумісним для клієнтів *Memcached*. *Couchbase* має наступний принцип роботи: при додаванні об’єкту до кешу, він розміщується в оперативній пам’яті, як і у випадку з *Memcached*, а його копія зберігається на диску. При переповненні кешу, розміщеного в оперативній пам’яті, елементи з нього видаляються, при цьому їх копії на жорсткому диску продовжують зберігатись, і в разі потреби клієнта кешу в даних, яких немає в оперативній пам’яті, вони завантажуються з жорсткого диску [5, 6, 7].

Однак, *Couchbase* має ті ж недоліки, які властиві *Memcached* – захаращення кешу об’єктами, які ще не «застаріли», проте ніким не використовуються, заповнення контейнеру ключів закешованих об’єктів «зайвими» ключами об’єктів, тощо.

Мета

Недослідженість питань, висвітлених у вступі визначили основну мету даної роботи, яка полягає у забезпеченні задовільного співвідношення якості обслуговування веб-серверів до собівартості в будь-який момент часу за рахунок оптимізації використання серверних ресурсів, мінімізуючи їх вартість шляхом прогнозування навантаження на сервер, а також у зменшенні часу відгуку веб-серверів за рахунок реалізації інтелектуального кешування даних.

Постановка та розв'язання задач

В даній роботі пропонується підхід, який, на нашу думку, зможе позбавити існуючі підходи до кешування об'єктів недоліків, пов'язаних з неефективним використанням оперативної пам'яті, які були наведені вище.

Основним критерієм ефективності кеша виступає швидкість реакції на запит закешованого об'єкту. Тобто, чим швидше ми одержимо об'єкт з кешу, тим вищою буде ефективність кешу. На нашу думку, підвищення ефективності кешу можна досягти за рахунок зменшення кількості закешованих об'єктів, які зберігатимуться в оперативній пам'яті до меж необхідності. Тобто, ми повинні визначити і залишити в кеші тільки ті об'єкти, в яких дійсно є потреба. Зменшення кількості закешованих об'єктів призведе до зменшення ключів, за якими буде здійснюватись пошук цих об'єктів, і відповідно призведе до підвищення швидкодії пошуку і зменшення часу, що витрачається на пошук ключа. Іншим позитивним ефектом, що досягається зменшенням кількості закешованих об'єктів, є більш ефективне використання оперативної пам'яті, через зменшення потреб в ній з боку кеша.

Ми будемо виходити з тих міркувань, що всі об'єкти в кеші є нерівномірно затребуваними на протязі заданого періоду часу, наприклад, доби. Тобто, одні об'єкти затребувані зранку, інші всередині дня, ті що залишились – ввечері. Крім того, можливо здійснити такий розподіл по годинах. Тому, доцільно буде зберігати закешовані об'єкти в оперативній пам'яті лише на протязі часового інтервалу, коли вони дійсно будуть потрібними, в інший же час, закешовані об'єкти можливо зберігати разом з їхніми ключами на диску.

Розглянемо принцип роботи такого механізму на прикладі одного закешованого об'єкту. При додаванні об'єкту до кеша, він розташовується в оперативній пам'яті, його ключ додається до набору ключів закешованих об'єктів, копія ключа і об'єкту переміщується на жорсткий диск. Далі, спеціальний програмний модуль починає відслідковувати актуальність закешованого об'єкту, яка полягає у кількості звернень до об'єкту в одиницю часу, наприклад, годину. Дані щодо актуальності закешованого об'єкту збираються на протязі керованого контрольного періоду часу. Інший програмний модуль прогнозування аналізує дані актуальності закешованого об'єкту і формує графік актуальності об'єкта в майбутньому. Графік являє собою таблицю, розподілену на часові інтервали і ознаку актуальності об'єкта у вказаний період – буде об'єкт актуальним чи ні. Приклад такої таблиці наведено в табл. 1.

Таблиця 1 – Актуальність «закешованого» об'єкту по часовим інтервалам

Межі часового інтервалу	Ознака актуальності «закешованого» об'єкту
12.00 – 13.00	+
13.00 – 14.00	-
…	…

Перед початком наступного часового інтервалу, спеціальний програмний модуль аналізує графік актуальності закешованих об'єктів і утворює нову множину ключів закешованих об'єктів, а також множину актуальних закешованих об'єктів, які розміщує в оперативній пам'яті на початку наступного часового інтервалу на час цього інтервалу. Попередні значення видаляються з оперативної пам'яті. Якщо для закешованого об'єкта закінчився термін життя в кеші – він видаляється автоматично і з оперативної пам'яті і з жорсткого диску. Кількість, тривалість часових інтервалів можуть бути налаштовані окремо з метою найбільш ефективного пристосування до умов роботи кеша. При збиранні даних, щодо «актуальності» закешованого об'єкта, якщо об'єкт є неактуальним (незатребуваним), навіть, якщо він не застарів, він може бути видалений як з оперативної пам'яті, так і з жорсткого диску до настання того моменту, коли він дійсно застаріє.

Реалізацію даного модуля пропонується здійснити на основі технології ідентифікації часових рядів (ЧР) з використанням нейронних мереж радіально-базисного типу. Метою прогнозування часових рядів є оцінка значень ЧР в майбутньому, що базується на даних, які були отримані в минулому та поточний момент часу. Схематично даний підхід до кешування даних зображений на рисунку 2.

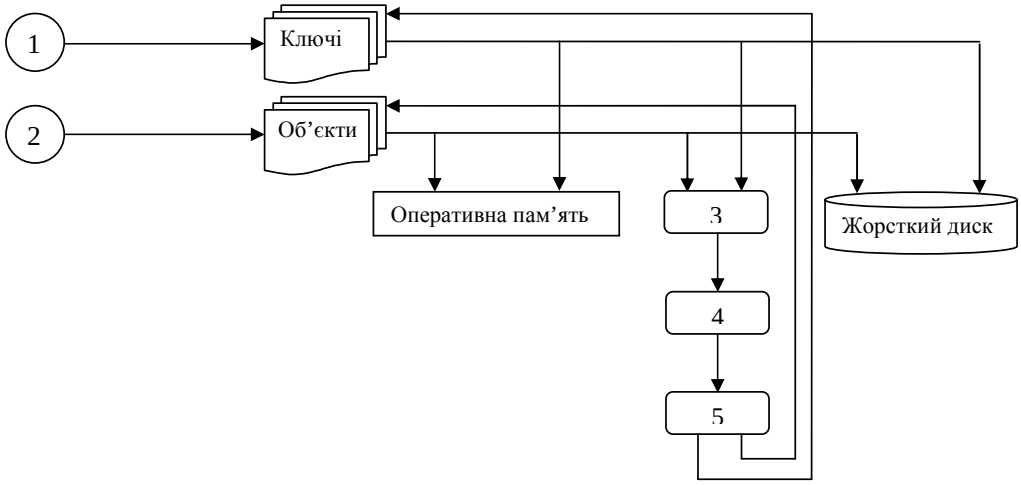
Математично ЧР являє собою ряд динаміки, впорядкований за часом, або сукупність спостережень деякої величини в різні моменти часу:

$$x(t) = \{x(t_1), x(t_2), x(t_3), ..., x(t_N)\}, \quad (1)$$

де t_1, t_2,…,t_N – моменти спостережень, N – довжина часового ряду.

Вибір нейронних мереж радіально-базисного типу для розв'язання даної задачі пояснюється тим, що вони дозволять з легкістю досягти компромісу високої точності прогнозу та малих витрат часу на

його здійснення, що надає можливість застосовувати їх в режимі реального часу для надання коротко-строкових прогнозів.



1 – ключ «закешованого» об’єкта,
2 – «закешований» об’єкт,
3 – модуль, що відслідковує і збирає дані щодо актуальності «закешованого» об’єкта,
4 – модуль, що аналізує дані актуальності «закешованого» об’єкта і здійснює прогнозування,
5 – модуль, що на основі даних, отриманих від модуля 4, управляє часом життя «закешованого» об’єкта в оперативній пам’яті.

Рисунок 2 – Схема інтелектуального кешування даних

Оскільки результатом модуля прогнозування для окремого є значення бінарного типу, то поставлена задача зводиться до задачі класифікації і пропонується використовувати ймовірнісну нейронну мережу (PNN - Probabilistic Neural Network) [8]. Архітектура ймовірнісної нейронної мережі зображена на рис. 3. Особливістю мереж даного типу є те, що їх структура формується в ході навчання. Розмірність N векторів навчальної вибірки визначає число нейронів і структуру вхідного шару ймовірнісної нейронної мережі. Загальний розмір навчальної вибірки відповідає загальній кількості нейронів шару зразків. Пред’явлення мережі кожного з зразків навчальної вибірки супроводжується вказівкою від вчителя номера j-го класу, якому належить вхідний зразок. Після пред’явлення всіх векторів навчальної вибірки, формується структура мережі, і стають визначеними параметри мережі у вигляді матриці. На цьому процес навчання ймовірнісної нейронної мережі завершується і мережа готова до класифікації невідомих зразків.

В класичній PNN кожний нейрон шару сумування сумує рівні активності всіх нейронів шару зразків свого класу і видає на своєму виході загальний рівень активності даного j -го класу. Далі визначається який нейрон шару сумування має максимальний вихідний сигнал, тим самим (за номером j-го нейрона) визначається номер класу j, до якого з більшою ймовірністю належить пред’явлений образ X.

Функція активності j-го нейрона сумування визначає значення густини розподілу ймовірностей для всього j -го класу:

$$\hat{f}_{n_j}(x) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_{n_j}\left(x, X_i^{(j)}\right), \qquad (2)$$

де n_j – кількість зразків в класі j; $j=1,\ldots,M$; K – ядерна функція.

Тоді дискримінантна функція оцінки буде обчислюватись за формулою:

$$\hat{d}_{n_j}(x) = \frac{1}{n}\sum_{i=1}^{n_j} K_{n_j}\left(x, X_i^{(j)}\right). \quad (3)$$

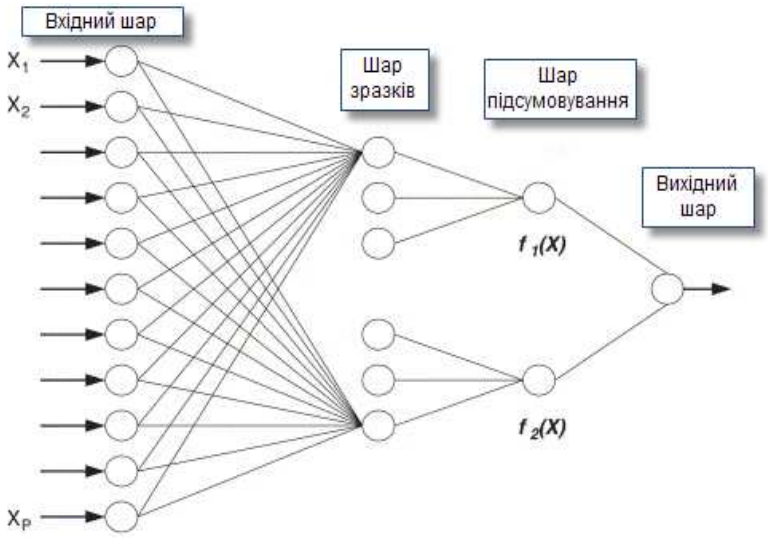


Рисунок 3 – Архітектура PNN

Відповідна процедура класифікації:

$$\hat{\varphi}_n(x) = m, \quad \text{ÿêùî} \sum_{i=1}^{n_m} K_{n_m}\left(x, X_i^{(m)}\right) \geq \sum_{i=1}^{n_j} K_{n_j}\left(x, X_i^{(j)}\right) \quad (4)$$
$$\text{äëÿ } i \neq m, i = 1,..., M.$$

Модуль прогнозування актуальності об’єкта буде являти собою багатоетапну технологію прогнозування на основі PNN, яка включатиме в себе реалізацію накопичення та попередньої обробки данних, формування навчальних зразків, навчання НМ, пошук розв’язку для визначення актуальності заданого об’єкту.

Отже, запропонований підхід дасть змогу оптимізувати кеш за рахунок зберігання в ньому лише дійсно необхідних об’єктів і їх ключів, що, в свою чергу, підвищить його швидкодію за рахунок зменшення часу на пошук та одержання об’єкту.

Для моделювання ефективності функціонування пулу серверів є необхідним ввести показники, за допомогою яких можна було б здійнити її оцінку. Необхідна потужність – рівень навантаження на пул ресурсів, який відображає максимальну потребу в обчислювальній потужності. Необхідна потужність має завжди задовольняти наступну умову: «сумма максимальнх навантажень на кожний ресурс не має перевищувати потужність серверу».

Навантаження на веб-сервери може бути визначене величиною параметрів, які відповідають обсягам запитів та з’єднань до них. Практичний досвід та результати вказують на складний вигляд процесу зміни вказаних параметрів. Було доведено, що для веб-серверів типовими є як стаціонарні, так і нестаціонарні режими експлуатації з багатоперіодичним характером зміни величин функціональних параметрів. Нестаціонарність режимів експлуатації проявляться у нестаціонарності моментів виникнення/зникнення періодичних складових [2, 9].

Для розв’язання задачі короткострокового прогнозування навантаження на сервер пропонується використати узагальнено-регресійну нейронну мережу GRNN (General Regression Neural Network). Архітектура даної мережі подібна до PNN. Прогнозування навантаження на сервер з використанням GRNN буде являти собою багатоетапну технологію, яка включатиме в себе: накопичення данних; попередню

обробку даних; формування навчальних зразків; навчання НМ; пошук розв’язку для визначення рівня навантаження на сервер в майбутньому.

Висновки

В роботі здійснено аналіз підходів до підвищення якості обслуговування веб-серверів. Виділені недоліки існуючих методів підвищення QoS, які базуються на технології кешування даних та прогнозуванні навантаження на сервер.

Запропоновано підхід інтелектуального кешування даних на основі прогнозування актуальності закешованих об’єктів з використанням нейронної мережі радіально-базисного типу, який на відміну від існуючих, реалізує оптимізацію кешу за рахунок зберігання в ньому лише дійсно необхідних об’єктів і їх ключів, що, в свою чергу, підвищує його швидкодію.

Для прогнозування навантаження на веб-сервер запропоновано використання інформаційної технології прогнозування часових рядів на основі узагальнено-регресійної нейронної мережі. Є доцільним продовження даних досліджень і здійснення перевірки ефективності запропонованих підходів на реальних даних.

Список літератури

1. Sravanthi Kalepu, Shonali Krishnaswamy, Seng Wai Loke, Verity: A QoS Metric for Selecting Web Services and Providers [Електронний ресурс] // Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW’03). – 2004. – Режим доступу до файлу: http://www.acs.org.au/vic/socsig/IEEE-VeritySOA-Krishnaswamy.pdf

2. Терейковська Л.О. Архітектура марківської моделі зміни навантаження Web-сервера / Л.О. Терейковська // Вісник ДУІКТ. – 2012. – Т.10, №1. – С. 95-100.

3. Tom Vercauteren, Pradeep Aggarwal, Xiaodong Wang Hierarchical Forecasting of Web Server Workload Using Sequential Monte Carlo Training // IEEE transactions on signal processing. – 2007. – VOL. 55. – № 4. – PP. 635-644.

4. Memcached [Електронний ресурс]. – Режим доступу до файлу: http://en.wikipedia.org/wiki/Memcached

5. Membase Server is Now Couchbase Server [Електронний ресурс]. – Режим доступу до файлу: http://www.couchbase.com/membase.

6. Disk Storage [Електронний ресурс]. – Режим доступу до файлу: http://www.couchbase.com/docs/ couchbase-manual-1.8/couchbase-introduction-architecture-diskstorage.html.

7. Couchbase features [Електронний ресурс]. – Режим доступу до файлу: http://www.couchbase.com/couchbase-server/features.

8. Квєтний Р.Н. Імовірнісні нейронні мережі в задачах ідентифікації часових рядів / В. В. Кабачій, О. О. Чумаченко [Елеткронний ресурс]. – Режим доступу до файлу.: http://www.nbuv.gov.ua/e-journals/vntu/2010_3/2010-3.files/uk/10rnktsi_ua.pdf.

9. Agustín C. Caminero, Salvador Ros, Roberto Hernández, Antonio Robles-Gómez, Rafael Pastor Cloud-based e-Learning Infrastructures with Load Forecasting Mechanism Based on Exponential Smoothing: A Use Case [Елеткронний ресурс]. – Режим доступу до журн.: http://fie-conference.org/fie2011/papers/1085.pdf.

Стаття надійшла: 14.10.2012.

Відомості про авторів

Квєтний Роман Наумович – д.т.н., проф., завідувач кафедри АІВТ, Вінницький національний технічний університет, (0432)598243, м. Вінниця, вул. Хмельницьке шосе 95.

Дружиніна Ольга Олегівна – аспірант кафедри АІВТ, Вінницький національний технічний університет, (0432) 598243, oo.druzhinina@gmail.com, м. Вінниця, вул. Хмельницьке шосе 95,