УДК 004.021

Є.В. Крилов, В.К. Анікін, А.О. Шумада

ОПТИМІЗАЦІЯ РОБОТИ ВЕБ-СЕРВЕРА ДЛЯ ВЗАЄМОПОВ'ЯЗАНИХ ПРОЦЕСІВ

Анотація: стаття присвячена питанням побудови планувальника взаємопов'язаних процесів. Досліджені особливості існуючих систем реального часу і їх проблеми при застосуванні до веб-серверів. Запропоновані критерії якості оцінки планувальника на основі часу виконання процесів. Побудована оптимальніша система планування процесів і перевірена її працездатність на прикладі.

Ключові слова: Ключові слова: карусельна диспетчеризація, системи реального часу, взаємопов'язані процеси.

Вступ

На сьогоднішній день однією з основних вимог до веб-сайтів є його швидкодія, адже час завантаження сторінки може сильно вплинути на враження користувача від нього. На швидкість завантаження вашого сайту можуть впливати кілька чинників: якість розробки, що включає як грамотність написання коду та верстки, недозоване та неграмотне використання анімаційних елементів, нефахова підготовка та розміщення матеріалів наповнення веб-сайту, в першу чергу це стосується зображень та фотоматеріалів. Значною мірою це стосується і завантаження основної HTML сторінки, яка може містити значні об'єми даних.

Нашою метою є зменшення часу відповіді сервера для кожного з користувачів.

В даній статті веб-сервер розглядається як система реального часу, яка обробляє зв'язані між собою N програм, які в свою чергу виконують M процесів. Нашою задачею буде створення оптимального планувальника процесів.

Огляд існуючих алгоритмів планування процесів

Існуючі алгоритми планування [1;2] можна розділити на дві категорії відповідно до їх поведінки після переривань: з перемиканням, та без перемиканням. Очевидно, що для задач веб-сервера недопустима монополізація використання процесорного часу, тому для наших цілей виберемо планування з перемиканням.

В таких системах важливим є коефіцієнт використання процесорного часу системи періодичних завдань, що є сумою відносин часу виконання кожного завдання до її періоду:

$$U = \sum u = \sum \frac{e}{p}, \tag{1}$$

де e – час виконання процесу, p – період виконання процесу, u – коефіцієнт використання процесорного часу періодичним процесом.

 ISSN 1560-8956

Для процесів, які не можуть бути розплановані жодним алгоритмом замість поняття "коефіцієнт використання процесорного часу" використовують поняття "щільність δ процесу", який визначається як

$$\delta = \frac{e}{\min(D, p)}, \tag{2}$$

де D – відносний крайній термін. Він в свою чергу визначається як

$$D = d - r, \tag{3}$$

де d – абсолютний крайній термін, до якого потрібно виконати програму, а r – початок процесу.

В класичній теорії систем реального часу програми можна переривати після використання кванту часу для надання можливості виконанню більш пріоритетному завданню.

При великому навантаженню на сайт існує велика ймовірність того, що конкуруючим запитам потрібні результати виконання проміжних процесів, наприклад отримання даних для відображення на головній сторінці, при цьому інші проміжні задачі, які можуть не співпадати, наприклад формування розмітки одного з блоків веб-сторінки. При такій ситуації необхідно надавати найбільший пріоритет процесам запиту до бази даних.

При цьому слід не забувати про людину, яка очікує на відповідь від сервера. Тому необхідно встановити деякий граничний максимальний час виконання запиту для одного користувача.

Резервом (запасом) часу $(L(t))$ називається різниця між часом, що залишився до крайнього терміну і часом, який завданню ще потрібно опрацювати, тобто

$$L(t) = (d - t) - e, \tag{4}$$

де t – поточний момент часу.

Визначимо базову схему призначення пріоритетів. До таких належать: "Турнірне визначення пріоритету", "Round robin", "Найкоротша завдання – перша", "Найменший час, що залишився для виконання", "Адаптивна диспетчеризація", "Карусельна диспетчеризація" [2].

Для нашої задачі найкраще підходить карусельна диспетчеризація. При карусельної диспетчеризації процес продовжує виконання, поки не настане момент, коли він:

- добровільно поступається управління (тобто блокується);
- витісняється процесом з вищим пріоритетом;
- використав свій квант часу (timeslice).

Після того, як процес використав свій квант часу, управління передається наступному процесу, який знаходиться в стані готовності і має такий же, або менший рівень пріоритету, якщо рівного немає.

Даний алгоритм не враховує такі фактори як залежності між процесами і максимальний час виконання запиту (програми). Тому в даній статті я пропоную модифікацію даного методу для використання в веб-серверах для вирішення задачі мінімізації часу виконання N процесів.

Постановка задачі

Нехай у нас є N програм, кожна з яких складається з M взаємопов'язаних процесів. Матриця пріоритетів процесів буде виглядати наступним чином:

$$\begin{pmatrix} K_{11} & \dots & K_{1M} \\ & \ddots & \\ \vdots & K_{ij} & \vdots \\ & \ddots & \\ K_{N1} & \dots & K_{NM} \end{pmatrix}. \tag{5}$$

Кожна процес P_{ij} має певний пріоритет K_{ij} і час виконання T_{ij}. Пов'язаність процесів P задана матрицею зв'язаності S:

$$\begin{array}{c|ccccc} & P_{11} & P_{1M} & \dots & P_{1N} & P_{NM} \\ \hline P_{11} & S_{11} & & \dots & & S_{1F} \\ P_{1M} & & \dots & & \dots & \\ \dots & & & S_{ab} & & \\ P_{N1} & & \dots & & \dots & \\ P_{NM} & S_{E1} & & \dots & & S_{EF} \end{array}, \tag{6}$$

де E і F – сумарна кількість процесів всіх програм, за однакової кількості процесів в програмах $E = F = N * M$, a і b – індекси в матриці зв'язаності, які відповідають конкретним процесам P_{ij}, $a = [1..E]$, $b = [1..F]$.

$$S_{ab} = \begin{cases} 0, & \text{якщо процес } P_{ij} \text{ не зв'язаний з іншим процесом } P_{ij} \\ 1, & \text{якщо процес } P_{ij} \text{ зв'язаний з іншим процесом } P_{ij} \end{cases} \tag{7}$$

Таким чином в системі існують процеси, результат яких може бути використаний іншим потоком. За наявності конкуруючих потоків потрібно не допустити повторного паралельного виконання процесів. Для цього потрібно використати метод вирівнювання пріоритетів для зв'язаних процесів.

Для всіх зв'язаних процесів ($S_{ab} = 1$), перед якими виконуються процеси з меншим пріоритетом, піднімемо пріоритет:

$$K_{ij} = K_{\max l} + 1, \tag{8}$$

де $K_{\max l}$ – максимальний пріоритет пов'язаних процесів.

Зменшуючи час відповіді сервера для одного користувача можна сильно збільшити час очікування для іншого користувача. Враховуючи той факт, що користувачі для нас мають однакову важливість, необхідно ввести ще один критерій оптимальності – різниця між часом відповіді сервера для різних користувачів (потоків). Значення цього показника вирахуємо через середньоквадратичне відхилення (q).

$$q = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}, \tag{9}$$

де N – кількість програм, x_i – час виконання однієї програми, $\bar{x}$ – середній час виконання програми.

Цей показник тоді кращий, коли середньоквадратичне відхилення менше.

Отримуємо оптимізований алгоритм вибору процесу для виконання на кожному кроці:

1. Визначаємо зв'язані процеси
2. Застосовуємо метод вирівнювання пріоритетів (8)
3. Припускаємо виключення з планувальника вирішених процесів іншим потоком
4. Перевіряємо показники якості (9) і обмеження (4)
5. Якщо показники якості покращились і при цьому не порушуються накладені обмеження, то фіксуємо кроки 2-3, інакше відміняємо ці кроки
6. Вибираємо процес для виконання з найбільшим пріоритетом. Якщо минулий квант часу виконувався процес з цього потоку, при цьому існує процес з таким же пріоритетом, віддаємо йому право на виконання

Експериментально-дослідне вирішення задачі

Основною характеристикою швидкодії веб-сервера з точки зору кінцевого користувача є його час відповіді. Тому сервер буде тоді оптимальніший, коли час відповіді для кожного користувача буде найменшим, тобто час виконання всіх програм буде найменшим. Таким показником якості є час виконання останнього процесу в останній програмі $(T3)$.

Нехай існує веб сервер, який одночасно обробляє $N = 3$ запити (програми). В кожній програмі йому треба виконати $M = 3$ процеси: аутентифікувати користувача (процес 1), отримати загальну

інформацію з бази даних (процес 2), сформувати відповідь (процес 3). наступна система потоків із процесами, які мають наступні статичні пріоритети та тривалість у форматі “пріоритет ; тривалість”:

Таблиця 1

Початкові пріоритети і тривалості процесів

	Програми		
Процеси	1	2	3
1	1;2	1;2	1;2
3	3;2	3;2	3;2
2	2;2	2;2	2;2

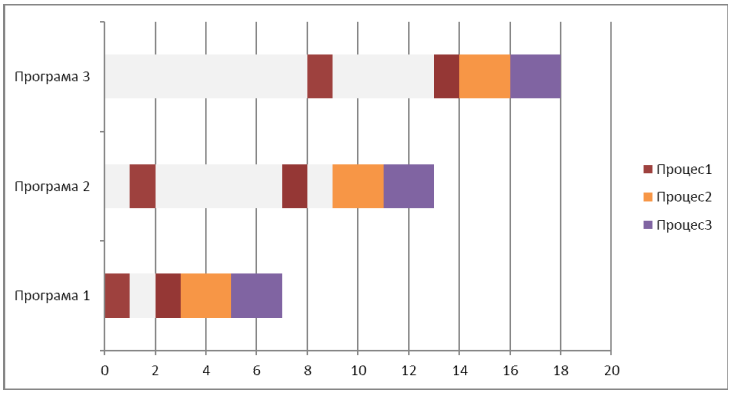

Рис. 1 – Часова діаграма роботи системи за методом карусельної диспетчеризації

Час виконання алгоритму – 18. Підрахуємо показник якості:

$$q = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} = 6.05. \qquad (10)$$

Процес 2 хоч і має більший пріоритет ніж процес 1, але в програмах 2 і 3 він змушений чекати закінчення виконання менш пріоритетних процесів інших потоків, оскільки процес, від якої вона залежить не може бути виконана. Побудуємо матрицю зв’язності S:

 ISSN 1560-8956

$$S = \begin{matrix} 1 & - & - & - & - & - & - & - & - \\ - & 1 & - & - & 1 & - & - & 1 & - \\ - & - & 1 & - & - & - & - & - & - \\ - & - & - & 1 & - & - & - & - & - \\ - & 1 & - & - & 1 & - & - & 1 & - \\ - & - & - & - & - & 1 & - & - & - \\ - & - & - & - & - & - & 1 & - & - \\ - & 1 & - & - & 1 & - & - & 1 & - \\ - & - & - & - & - & - & - & - & 1 \end{matrix}. \qquad (11)$$

Застосуємо запропонований алгоритм і порівняємо показник якості системи.

Таблиця 2

Пріоритети і тривалості процесів після застосування запропонованого алгоритму

	Програми		
Процеси	1	2	3
1	4;2	3;2	3;2
3	4;2	3;0	3;0
2	2;2	2;2	2;2

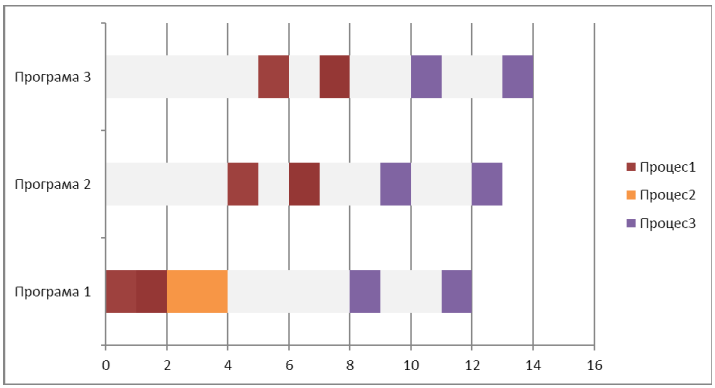

Рис. 2 – Часова діаграма роботи системи за запропонованим алгоритмом

Час виконання алгоритму – 14. Підрахуємо показник якості:

$$q = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} = 0.9. \qquad (12)$$

Висновки

1. В цій статті розглянуто необхідність створення планувальника для веб-сервера.
2. Запропоновано використання методу карусельної диспетчеризації і здійснено його модифікацію.
3. Проведене дослідження, яке показало ефективність модифікованого методу, який покращує показники роботи веб-сервера на 28% та на 572%.

Список використаних джерел

1. *Климентьев К.Е.* Системы реального времени / Климентьев К.Е. – Самара, 2008. – С. 21–22.
2. Егоров В.Ю. Новые подходы к диспетчеризации задач в операционных системах. / В.Ю. Егоров // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2008 – Вып. 2

Отримано 28.03.2014 р.