УДК 004.451

Ю.П. КЛЬОЦ, С.В. МОСТОВИЙ
Хмельницький національний університет

ОЦІНКА ЧАСОВОЇ СКЛАДНОСТІ МЕТОДУ ПРОГНОЗУВАННЯ ВЗАЄМОБЛОКУВАНЬ ПРОЦЕСІВ В КОМП'ЮТЕРНИХ СИСТЕМАХ

В роботі проведено оцінку часової складності та ефективності методу прогнозування взаємоблокувань процесів в КС на основі експериментального дослідження на прикладі модифікованої СКБД MySQL. Отримані результати вказують на зменшення часових витрат на виконання процесів, що в свою чергу дозволяє опрацьовувати в 2,3 разу більшу кількість даних в одиницю часу для задач, в яких виникають взаємоблокування.

Ключові слова: взаємоблокування процесів, метод прогнозування взаємоблокувань.

Y.P. KLOTS, S.V. MOSTOVUY
Khmelnytsky National University

EVALUATION OF TIME COMPLEXITY OF METHOD OF PREDICTING THE DEADLOCK OF PROCESSES IN COMPUTER SYSTEMS

Abstract – In the paper, the estimation of time complexity and efficiency of method of predicting the deadlock of processes in a computer system based on the solution of the deadlock processes in the database management system MySQL, are given.

To test the method on the server to perform run PHP script that the parallel execution causes deadlocks. The study compared the runtime phase transaction with a maximum load of the system and the various modes.

The results obtained indicate a decrease in time spent on the execution of processes using the method of predicting deadlocks, which in turn allows you to process 2.3 times more data per unit of time for tasks in which there are often deadlocks.

Keywords: deadlock, method of forecasting of deadlock

Вступ

В процесі експлуатації комп'ютерних систем (КС) досить часто виникає ситуація блокування процесів, що виконуються у них [1].

Частковим випадком блокування процесів є можливість їх взаємного блокування. Взаємне блокування (англ. deadlock) – ситуація в багатозадачному середовищі або системах керування базами даних (СКБД), при якій кілька процесів перебувають у стані нескінченного очікування ресурсів, зайнятих самими цими процесами. Виникнення взаємних блокувань процесів призводить до збільшення часу їхнього виконання (може зростати до нескінченості), до не ефективного використання ресурсів КС (порожні цикли очікування).

Причинами даного явища є помилки у коді програмного забезпечення (ПЗ), невідповідність ПЗ його специфікації.

Для визначення і усунення взаємоблокування процесів у комп'ютерних системах розроблено ряд методів і засобів [1–3]. Проте на практиці їх не завжди доцільно застосовувати для вирішення задачі взаємоблокування, оскільки частина з них носить лише теоретичний характер і не може бути реалізованою в сучасних операційних системах [1, 4, 5]. Інша частина при реалізації стає достатньо громіздкою і ресурсоємною. Тому розробники сучасних ОС сімейств Windows та Unix, а також розробники сучасних СКБД не включають відомі алгоритми уникнення взаємоблокувань процесів.

Постановка задачі

За умови виконання в системі невеликої кількості процесів відсутність таких засобів була допустима. Проте стрімкий розвиток апаратних засобів, зростання об'єму та складності ПЗ, яке займається вирішенням масштабних та відповідальних задач, не повинно дозволяти виникнення взаємоблокування, що в свою чергу вимагає розробки нових підходів до вирішення цієї задачі.

Для усунення суттєвих недоліків відомих методів та алгоритмів вирішення проблеми взаємоблокування розроблено метод прогнозування стану взаємоблокування процесів, що враховує їхній життєвий шлях [6, 7] та відповідні алгоритми і засоби. Необхідно провести оцінку часової складності розробленого методу та відповідних йому засобів.

Основна частина

Процес – це система дій, що реалізує певну функцію в обчислювальній системі й оформлена так, що керуюча програма обчислювальної системи може перерозподіляти ресурси цієї системи з метою забезпечення багатозадачності [1]. Позначимо множину виконуваних процесів, як $A = \{a_i\}_{i=1}^{y}$, де y – кількість процесів.

Ресурс комп'ютерної системи – засіб комп'ютерної системи, що може бути виділений процесу обробки даних на певний інтервал часу. Позначимо множину наявних ресурсів КС, як $RE = \{re_j\}_{j=1}^{x}$, де x – кількість видів ресурсів.

До ресурсів КС віднесемо наявну пам'ять, процесори, пристрої вводу/виводу, засоби взаємовиключення (м'ютекси, семафори та ін.), а також дані, необхідні для роботи процесів (файли в пам'яті

та на зовнішніх носіях, результати обчислень інших процесів).

Кожен процес від моменту створення до моменту завершення проходить через ряд станів (рис. 1).

Рис. 1. Діаграма станів процесу, що включає стан взаємоблокування

Під сигнатурою процесу будемо розуміти сукупність його характеристик, яка однозначно ідентифікує стан процесу в КС в певний момент часу *t*:

$$a_i(t) \to (a_{i_1}^t, a_{i_2}^t, ..., a_{i_z}^t), \quad (1)$$

де $a_i(t) \in A$ – поточний процес, $a_{i_1}^t, \ldots, a_{i_z}^t$ – характеристики процесу в поточний момент часу (параметри та ресурси, які використовує процес в даний момент).

До характеристик процесу, що формують сигнатуру, віднесемо наступні: ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача, якому належить процес, пріоритет процесу, квоти процесу (кількість пам'яті і процесорний час доступні процесу), перелік отриманих ресурсів.

Оскільки до складу сигнатури процесу входять його унікальні характеристики, то в один і той самий момент часу в системі не існує двох абсолютно однакових сигнатур.

Отже, життєвий цикл процесу можна подати у вигляді послідовності станів, через які проходить цей процес. Перехід із стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з ряду причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому.

Позначимо стан процесу через *w*, причину зміни параметрів через *r*, а перехід із стану в стан через $w_i \xrightarrow{\text{зміна параметрів з причини } r_j} w_{i+1}$. Тоді життєвий цикл процесу буде мати вигляд (2):

$$a_i : w_0 \xrightarrow{r_j} w_1 \xrightarrow{r_j} \ldots \xrightarrow{r_j} w_{k-1} \xrightarrow{r_j} w_k, \quad (2)$$

де $w_0 \in W$ – початковий стан процесу (стан "створений"), $w_k \in W$ – кінцевий стан процесу (стан "завершений"), W – множина програмних станів процесу (рис.1), $r_j \in R$ – множина можливих причин зміни параметрів процесу (*j*=1,2,3…).

Враховуючи визначення сигнатури та (1) і (2), життєвий цикл кожного процесу можна подати у вигляді:

$$a_i : (a_{i_1}^{t_0}, a_{i_2}^{t_0}, ..., a_{i_z}^{t_0}) \xrightarrow{r_j} (a_{i_1}^{t_1}, a_{i_2}^{t_1}, ..., a_{i_z}^{t_1}) \xrightarrow{r_j} \ldots \xrightarrow{r_j} (a_{i_1}^{t_k}, a_{i_2}^{t_k}, ..., a_{i_z}^{t_k}), \quad (3)$$

У стан взаємоблокування можуть потрапляти процеси, що взаємодіють між собою у багатозадачних КС в певний момент часу. До потрапляння у стан взаємоблокування процеси протягом свого життєвого циклу знаходяться в інших станах, а саме стан "створений", стан "очікуючий", стан "виконуваний", стан "заблокований", стан "завершений" (рис.1). У стан взаємоблокування процеси потрапляють, як правило, із стану "заблокований". Отже, у даний момент часу серед множини процесів можна виділити підмножину процесів, які можуть в наступний момент часу потрапити до стану взаємоблокування. Перед входженням у стан взаємоблокування процес буде знаходитись у певному "граничному" стані [7], після якого ймовірність переходу у стан взаємоблокування буде високою. Взаємоблокування процесів призводить до часткової або повної втрати функційної здатності КС. Тому вважатимемо стан взаємоблокування процесів неробочим станом КС, а інші стани процесів – робочим станом КС.

Віднесемо до робочого стану такі стани процесу: стан "створений", стан "виконуваний", стан "завершений" та стан "очікуючий". До "граничного" стану віднесемо стан "заблокований".

Представимо шлях, яким процес потрапляє у стан взаємоблокування, у вигляді наступної схеми (рис.2).

Як видно із схеми, виникнення взаємоблокування процесів можливе лише для частини процесів, що знаходяться у граничному стані. При переході процесів до граничного стану відбувається зміна їхніх параметрів.

Розглянемо життєвий цикл процесу. В момент створення (стан "створений") процес знаходиться у

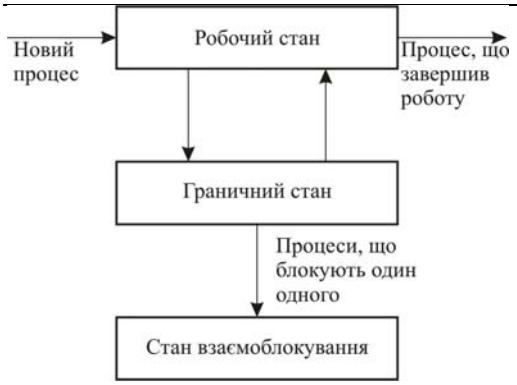

Рис. 2. Схема переходу процесів у стан взаємоблокування

робочому стані, йому надана частина ресурсів системи. Позначимо цей стан процесу, як $s_{\text{роб}}(t)$. В певний момент часу процесу для подальшого виконання необхідний додатковий ресурс системи, який на даний час є недоступний. Відбувається перехід процесу до стану "заблокований", тобто процес потрапляє у граничний стан. Позначимо цей стан процесу, як $s_{\text{гран}}(t)$, а перехід до даного стану, як $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$. При задоволенні потреб процесу у ресурсах він повертається у робочий стан, тобто здійснює перехід $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$. Коли необхідний ресурс отримати неможливо по причині його використання іншим процесом, то процес залишається у граничному стані. Якщо ж другий процес в свою чергу очікує ресурс, що зайнятий першим процесом, то вони обидва потрапляють у стан взаємоблокування. Позначимо цей стан процесу, як $s_{\text{взаємоблокування}}(t)$, а перехід до даного стану, як $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$.

Таким чином, враховуючи (3), життєвий цикл процесу буде мати один з наступних виглядів:

1) процес створений, йому надано всі необхідні для завершення роботи ресурси, він виконується і завершується (процес весь час знаходиться в робочому стані $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} s_1 \xrightarrow{r_j} s_k , \tag{4}$$

де $s_0 \in s_{\text{роб}}$, $s_1 \in s_{\text{роб}}$, $s_k \in s_{\text{роб}}$.

2) процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, через деякий час отримує їх, виконується і завершується (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, потім $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ знову в робочий стан $s_{\text{роб}}(t)$), тобто:

$$a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_k , \tag{5}$$

де $s_0 \in s_{\text{роб}}$, $s_l \in s_{\text{гран}}$, $s_k \in s_{\text{роб}}$.

3) процес створений, йому надано частину ресурсів, необхідних для завершення роботи, він потребує додаткових ресурсів, які утримує інший процес, який в свою чергу потребує ресурсів першого процесу (процес спочатку знаходиться в робочому стані $s_{\text{роб}}(t)$, потім переходить $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ в граничний стан $s_{\text{гран}}(t)$, із якого відбувається перехід $s_{\text{гран}}(t) \xrightarrow[\text{утримування ресурсу } re_j]{\text{очікування ресурсу } re_i} s_{\text{взаємоблокування}}(t)$ до стану взаємоблокування).

$$\begin{cases} a_i : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_l \xrightarrow{r_j} s_x \\ a_m : s_0 \xrightarrow{r_j} \ldots \xrightarrow{r_j} s_n \xrightarrow{r_j} s_x \end{cases}, \tag{6}$$

де $s_0 \in s_{\text{роб}}$, $s_l \in s_{\text{гран}}$, $s_n \in s_{\text{гран}}$, $s_x \in s_{\text{взаємоблокування}}$.

Із розглянутих вище можливих варіантів поведінки процесів критичним для КС є останній, при якому процеси потрапляють у стан взаємоблокування

Алгоритм прогнозування потрапляння процесів у стан взаємоблокування

Як видно із рис.2, прогнозування стану процесу включає два етапи: визначення множини процесів, що можуть потрапити у стан взаємоблокування; виділення із множини потенційних процесів групи процесів, що потраплять у стан взаємоблокування. Таким чином, алгоритм прогнозування стану процесу буде містити дві частини.

Згідно [6] до моделі прогнозування стану процесів входять наступні величини:

$$M = \langle A, S, D, P, R \rangle \tag{7}$$

де A – множина сигнатур процесів, що виконуються в КС у даний момент;

S – впорядкована послідовність характеристик комп'ютерної системи (загальний обсяг оперативної пам'яті, зовнішньої пам'яті, обсяг вільної в даний момент пам'яті, кількість периферійних пристроїв);

D – підмножина сигнатур процесів, що знаходяться у стані, наближеному до стану

взаємоблокування (у граничному стані);

P – множина правил, на основі яких визначається група процесів, що потраплять у стан взаємоблокування;

R - вектор ймовірностей переходу у стан взаємоблокування процесів із підмножини D.

Алгоритм 1. Виявлення потенційних процесів, що можуть потрапити у стан взаємоблокування (виявлення процесів, що знаходяться у граничному стані).

1. Якщо $a_k \in A$ потребує ресурс $r_i \in R$, то перехід до 2.

2. Якщо $a_k \in A$ знаходиться в стані $s_{\text{роб}}(t)$, то перехід до 3, інакше перехід до 4.

3. Виконати для $a_k \in A$ перехід із робочого стану до граничного $s_{\text{роб}}(t) \xrightarrow{\text{потреба ресурсу } re_i} s_{\text{гран}}(t)$ та включити $a_k \in A$ до $D \subset A$. Перехід до 4.

4. Якщо $a_k \in A$ надано у використання ресурс $re_i \in RE$, то перехід до 5, інакше перехід до 6.

5. Виконати для $a_k \in A$ перехід із граничного стану до робочого $s_{\text{гран}}(t) \xrightarrow{\text{надання ресурсу } re_i} s_{\text{роб}}(t)$ та виключити $a_k \in A$ із $D \subset A$. Перехід до 6.

6. Якщо перевірено всю множину А, то перехід до 7, інакше перехід до 1.

7. Кінець алгоритму.

Для визначення процесів, що потраплять у стан взаємоблокування, використовується система прогнозування стану процесів, в основі якої лежить система нечіткого логічного висновку (СНЛВ). На рис.3 показана структура СНЛВ.

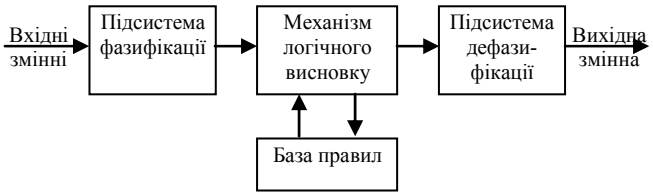

Рис. 3. Загальна схема СНЛВ

Підсистема фазифікації призначена для визначення ступеня приналежності вхідних значень $x_g \in X$, $X = D \cup S$, $g = \overline{1,h}$ до нечітких множин входу, що являють собою лінгвістичні змінні з відповідної лінгвістичної шкали $T_{x_g} = \left\{ T_{x_g}^1, T_{x_g}^2, ..., T_{x_g}^{m_{x_g}} \right\}$, де m_{x_g} - кількість лінгвістичних змінних у шкалі для g-го входу. Необхідність у фазифікації зумовлена тим, що у СНЛВ використовуються лінгвістичні правила.

База правил, що містить лінгвістичні правила, є основою механізму логічного висновку. Механізм логічного висновку здійснює відображення вхідних нечітких множин T_{x_g} за допомогою кожного правила у вихідну T_y з набору вихідних лінгвістичних змінних $T_y = \left\{ T_y^1, T_y^2, ..., T_y^{m_y} \right\}$. Правила із множина правил $P = \left\{ P_j \right\}, j = \overline{1,n}$, що міститься в базі правил, подані у наступному форматі:

$$P_j = \text{"якщо } x_1 \in T_{x_1} \text{ і } x_2 \in T_{x_2} \ldots \text{і } x_h \in T_{x_h}, \text{то } y_j \in T_y\text{"} \tag{8}$$

Вихідні нечіткі множини y_i кожного правила об'єднуються в одну нечітку множину висновку $\tilde{y}$. Після цього підсистема дефазифікації здійснює відображення нечіткої множини висновку $\tilde{y}$ у чітке число $\bar{y}$, яке буде результатом СНЛВ для заданих вхідних значень x_g.

Алгоритм 2. Виявлення процесів, що потраплять у стан взаємоблокування:

1. Проводимо нормування характеристик КС за наступною формулою:

$$P_n = 1 - \frac{P_d + P_m}{P_d \cdot P_m}, P_d \neq 0, \tag{9}$$

де P_n – черговий нормований показник;

P_d – поточне значення показника в КС;

P_m – максимальне значення показника в КС.

2. Для кожного $x_g \in X$, $X = D \cup S$, $g = \overline{1,h}$ визначаємо ступінь належності до нечітких множин входу (ступені істинності $\mu_g^j(x_g)$).

3. На основі ступенів істинності передумов $\mu_g^j(x_g)$ для кожного правила P_j, $j = \overline{1,n}$ розраховуємо ступінь його виконання α_j за формулою.

$$\alpha_j = \min\left(\mu_1^j(x_1), \mu_2^j(x_2), ..., \mu_h^j(x_h) \right) \tag{10}$$

4. На основі ступеню виконання α_j для кожного правила $P_j, j=\overline{1,n}$ розраховуємо результат його виконання.

5. На основі результату виконання кожного правила $P_j, j=\overline{1,n}$ визначаємо вихідну нечітку множину з усіченою функцією приналежності $\ddot{\mu}^j(y)$ за формулою:

$$\ddot{\mu}^j(y)=\min\left(\alpha_j,\mu^j(y)\right) \tag{11}$$

6. Вихідні нечіткі множини $\ddot{\mu}^j(y)$ згідно (12) агрегуємо в нечітку множину висновку $\tilde{y}$, що має функцію приналежності (13).

$$\tilde{y}=\max\left(\mu^j(y)\right), j=\overline{1,r} \tag{12}$$

$$\mu_{\tilde{y}}=\max\left(\ddot{\mu}^1(y),\ddot{\mu}^2(y),...,\ddot{\mu}^r(y)\right) \tag{13}$$

7. Приводимо до чіткості нечітку множину $\tilde{y}$ за допомогою процедури дефазифікації центроїдним методом

$$\overline{y}=\frac{\int_{C_1}^{C_t} x\cdot f_{\tilde{y}}(x)dx}{\int_{C_1}^{C_t} f_{\tilde{y}}(x)dx} \tag{14}$$

8. Встановлюємо для R_k значення $\overline{y}$.

9. Повторюємо кроки 1-8 k разів.

10. Кінець алгоритму.

Оцінка часової складності методу прогнозування взаємоблокування процесів

Для оцінки часової складності методу прогнозування взаємоблокувань процесів було використано web-сервер Apache2 з PHP 5 та sql-сервер MySQL. Для перевірки роботи методу на сервері запускався на виконання PHP-скрипт, представлений у лістингу 1, який при паралельному виконанні призводить до виникнення взаємоблокувань.

Лістинг 1

```
$sql = "SELECT COUNT(*) FROM t "; $x = mysql_query($sql);
$r = mysql_fetch_row($x);$max_id = $r[0];
$id1 = rand(0,$max_id); do { $id2 = rand(0,$max_id); } while ($id1==$id2);
mysql_query("START TRANSACTION;");
$sql1 = "select a from t where id = $id1 for update"; mysql_query($sql1);
//Вибір кортежу для обрахунку
usleep(100); //моделювання обробки даних
$sql2 = "select b from t where id = $id2 for update"; mysql_query($sql2);
mysql_query("COMMIT;"))
```

В представленому коді видалено бізнес-логіку, зате повністю збережено послідовність запитів, що при паралельному виконанні призводять до появи взаємних блокувань.

Рівні взаємоблокувань, представлені на рис.4, показують взаємоблокування в КС. Суцільною лінією показано рівень взаємоблокувань при використанні стандартних засобів MySQL для виявлення заблокованих трансакцій. Пунктирною лінією показано рівень взаємоблокувань, що виникали при використанні запропонованого методу прогнозування взаємоблокувань.

Час виконання процесів при використанні різних механізмів вирішення взаємоблокувань показаний на рис.5. Суцільною лінією показано середній час виконання процесів при використанні стандартного механізму вирішення взаємоблокувань. Пунктирною – середній час виконання процесів при використанні запропонованого методу прогнозування взаємоблокувань. Штрих-пунктирною – середній час виконання процесів, за умови ненастання взаємоблокувань.

У випадку використання стандартного механізму виявлення взаємоблокувань спостерігається значне зростання часу виконання навіть при незначній кількості паралельних процесів. В цей самий період не спостерігається значного завантаження процесора (не більше 20%), оскільки процеси більшу частину часу очікують доступу до заблокованих ресурсів.

У випадку, коли не виникає взаємоблокувань процесів, час їх виконання зростає незначно (на 3% при кількості паралельних процесів 25 і рівні завантаження процесора не більше 20%).

Використання запропонованого методу прогнозування взаємоблокувань показує помітно пологіше наростання середнього часу виконання процесів, за таких самих умов.

Зниження часу виконання процесів при виконанні єдиного потоку пояснюється відсутністю затримок, пов’язаних з очікуванням звільнення ресурсу.

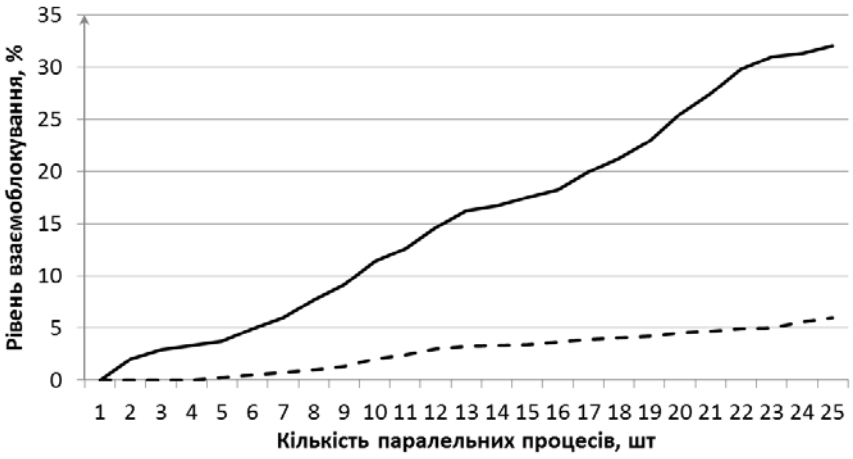

Рис. 4. Залежність рівня взаємоблокувань від кількості паралельних процесів

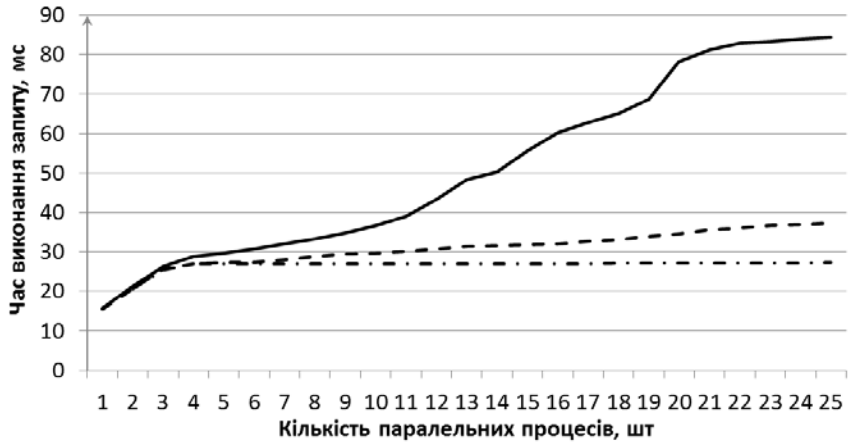

Рис. 5. Залежність часу обробки запитів від кількості паралельних процесів

В ході дослідження було проведено порівняння часу виконання фаз трансакцій при максимальному завантаженні системи і різних режимах роботи, що представлено на рис.6.

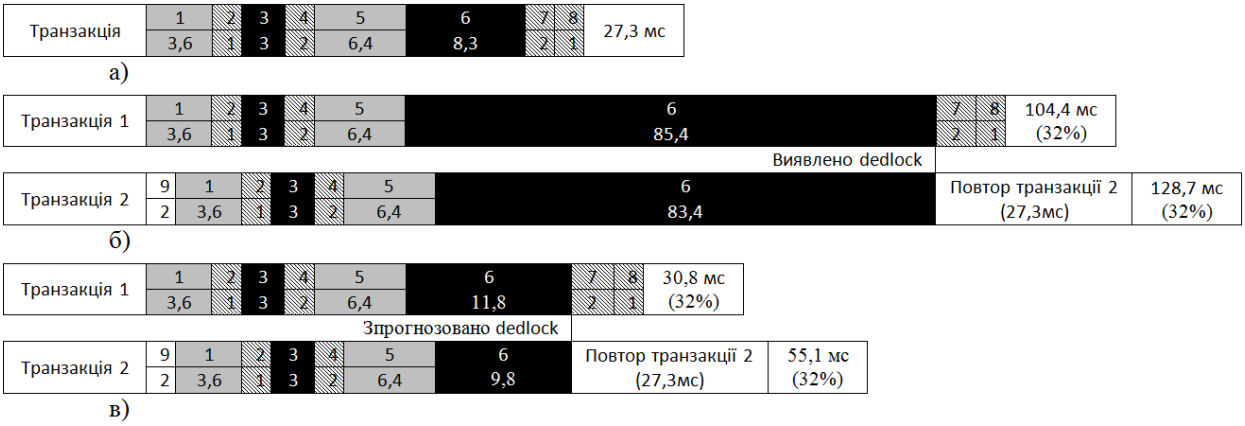

Рис. 6. Часові діаграми виконання фаз транзакції:
а) без взаємоблокування; б) взаємоблокування із стандартним виявленням; в) взаємоблокування із прогнозуванням.
(1 – Підготовчі дії; 2 – Запуск трансакції; 3 – Очікування звільнення кортежу id1; 4 – Виконання 1-го SELECT;
5 – Опрацювання даних; 6 – Очікування звільнення кортежу id2; 7 – Виконання 2-го SELECT;
8 – Завершення трансакції; 9 – Затримка від початку запуску)

На рис. 6.а представлена часова діаграма виконання фаз трансакції за умови, що при її виконанні не відбулось взаємоблокування. Середній час виконання трансакції склав 27,3 мс, з них 11,3 мс (41%) зайняло очікування заблокованих ресурсів. В ході дослідження таких трансакцій виявилось 36%.

На рис. 6.б представлено часову діаграму взаємоблокування двох трансакцій із стандартним механізмом їх вирішення. Трансакція 1 при першому звертанні (4) до таблиці БД вибирає та блокує кортеж з ключем id1. Трансакція 2 розпочинає виконуватись на 2 мс пізніше, ніж Трансакція 1 і при першому звертанні (4) вибирає та блокує запис із ключем id2. Після опрацювання отриманих даних (5) трансакція 1 звертається до кортежу з id2. Оскільки він заблокований, то трансакція переходить до очікування звільнення ресурсу (6). В свою чергу трансакція 2 звертається до кортежу з id1. Він також виявляється заблокованим і трансакція 2

переходить в режим очікування звільнення ресурсу (6). Процеси потрапляють в ситуацію взаємного блокування і самостійно не можуть вийти з циклу нескінченного очікування. Взаємоблокування вирішує КС періодично аналізуючи граф процесів і ресурсів. Оскільки задача є алгоритмічно складною, то вона виконується через інтервали часу в 150–200 мс. Після виявлення взаємоблокування один із процесів, що пізніше надійшов у систему, примусово завершується і повторно запускається на виконання. Інший з заблокованих процесів продовжує своє виконання. Великі часові проміжки між повторними аналізами графа призводять до значних затримок при виявленні взаємоблокувань. Середній час процесу, який був виконаний повторно складає 128,7 мс, процесу, який продовжив роботу після взаємного блокування – 104,4 мс. В ході дослідження таких трансакцій виявилось по 32%, оскільки процеси у взаємоблокування потрапляють парами.

На рис. 6.в представлено часову діаграму взаємоблокування двох трансакцій із прогнозуванням взаємоблокувань. До моменту запиту в трансакції 2 кортежу з ключем id2 часові діаграми ідентичні. В момент цього запиту (6) трансакція 2 потрапляє в граничний стан і для неї проводиться прогнозування. Час прогнозування складає близько 10 мс. В результаті прогнозу визначається ймовірність взаємоблокування процесів, після чого обирається один із них. Цей процес знімається з виконання і запускається повторно. Середній час виконання процесу, що був виконаний повторно, складає 55,1 мс; процесу, що продовжив роботу після взаємного блокування – 30,8 мс. В ході дослідження таких трансакцій виявилось по 32%.

З отриманих досліджень середній час виконання трансакцій із стандартним механізмом виявлення взаємоблокувань склав: $27,3 \cdot 0,36 + 104,4 \cdot 0,32 + 128,7 \cdot 0,32 = 84,4$ мс, із прогнозуванням взаємоблокувань: $27,3 \cdot 0,36 + 30,8 \cdot 0,32 + 55,1 \cdot 0,32 = 37,3$ мс.

Застосування методу прогнозування взаємоблокувань забезпечило зменшення часу виконання процесів в $\frac{84,4}{37,3} = 2,3$ рази. Це дозволяє більш ефективно використовувати ресурси КС.

Висновки

В роботі запропоновано оцінку часової складності та ефективності методу прогнозування взаємоблокувань процесів в КС. Проведено його експериментальне дослідження на прикладі модифікованої СКБД MySQL. Середній час виконання процесу зменшився з 84,4 мс при використанні стандартних засобів MySQL до 37,3 мс при використанні запропонованого методу. Отримані результати вказують на зменшення часових витрат на виконання процесів, що в свою чергу дозволяє опрацьовувати в 2,3 рази більшу кількість даних в одиницю часу для задач, в яких часто виникають взаємоблокування (відбувається конкурентна боротьба за ресурси).

Література

1. Савенко О.С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О.С. Савенко, Ю.П. Кльоц, С.В. Мостовий // Вісник ХНУ. – 2007. – Т. 1. – № 3. – С. 248–251.

2. Coffman E.G. System deadlocks / E.G. Coffman, M.J. Elphick, A. Shoshani. // Computing Surveys. – June 1971. – Vol. 3. – №. 2. – P. 67–78.

3. Isloor S.S. The Deadlock Problem: An Overview / S.S. Isloor, T.A. Marsland. // Computer. – 1980. – Vol. 13. – № 9. – P. 58–78.

4. Nima Kaveh. Deadlock detection in distribution object systems / Nima Kaveh, Wolfgang Emmerich // Software Engineering Notes. – September 2001. – Vol. 26. – №. 5. – P. 44–51.

5. Saddek Bensalem. Confirmation of deadlock potentials detected by runtime analysis / Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, Laurent Mounier // International Symposium on Software Testing and Analysis – 2006. – P. 41–50.

6. Савенко О.С. Модель прогнозування стану процесів в комп'ютерній системі / О.С. Савенко, С.В. Мостовий // Радіоелектронні і комп'ютерні системи. – 2008. – № 5 (32). – С. 109–115.

7. Савенко О.С. Система прогнозування стану процесів в персональному комп'ютері / О.С. Савенко, С.В. Мостовий // Збірник праць VIII міжнародної конференції. – К., 2008. – С. 308–314.

References

1. Savenko O.S. Doslidjennya ta analiz blokuvannya procesiv v kompyuterniy systemi / O.S. Savenko, Y.P. Klots, S.V. Mostovuy // Visnyk KHNU. – 2007. – Vol.1, №3. – Pages 248-251. [in Ukrainian]

2. E.G. Coffman. System deadlocks / E.G. Coffman, M.J. Elphick, A. Shoshani. // Computing Surveys. – June 1971. – Vol.3, No.2. – Pages 67 – 78.

3. S.S. Isloor. The Deadlock Problem: An Overview / S.S. Isloor, T.A. Marsland. // Computer. – 1980. – Vol 13, №9. - Pages. 58-78.

4. Nima Kaveh. Deadlock detection in distribution object systems / Nima Kaveh, Wolfgang Emmerich // Software Engineering Notes. – September 2001. – Vol.26, No.5. – Pages 44 – 51.

5. Saddek Bensalem. Confirmation of deadlock potentials detected by runtime analysis / Saddek Bensalem, Jean-Claude Fernandez, Klaus Havelund, Laurent Mounier // International Symposium on Software Testing and Analysis – 2006. – Pages 41 – 50.

6. Savenko O.S. Model prognozuvannya stany procesiv v kompyuterniy systemi / O.S. Savenko, S.V. Mostovuy // Radioelektronni I kompyuterni systemy. – 2008. – №5 (32). – Pages 109-115. [in Ukrainian]

7. Savenko O.S. Systema prognozuvannya stanu procesiv v personalnomu kompyuteri / O.S. Savenko, S.V. Mostovuy // Zbirnyk prac VIII mignarodnoy konferenciy. – Kyiv. – 14-17 may, 2008. – Pages 308-314. [in Ukrainian]

Рецензія/Peer review : 20.1.2014 р. Надрукована/Printed :6.2.2014 р.
Рецензент: д.т.н., проф. ОІМ, ХНУ, Шалапко Ю.І.