УДК 004.032.32:004.925

Т.С. Іващук
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

АЛГОРИТМ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ МУЛЬТИТЕРМІНАЛЬНОЇ СИСТЕМИ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Розглянуто алгоритм ефективного використання ресурсів мультитермінальної системи в режимі реального часу.

Ключові слова – мультитермінальна система, реальний час.

An algorithm of efficacious use of multiterminal system resources in real-time mode is considered

Keywords – multiterminal system, real time.

Вступ. Мультитермінальні системи вигідні як альтернатива звичайним комп'ютерним мережам. Їх використання є актуальним при створенні комп'ютерних класів та інших комп'ютеризованих приміщень, оскільки дає змогу зменшити грошові затрати. Для забезпечення комфортної роботи всім користувачам мультитермінальні системи повинні проводити вчасну обробку всіх запитів. Проте через використання обмежених ресурсів у цьому випадку необхідне максимально ефективне їх використання в режимі реального часу. Досягти цього можна за допомогою алгоритмів ефективної обробки запитів.

Огляд літературних джерел. У [1] розглянуто класифікацію та характеристики відомих алгоритмів обробки запитів. При цьому не зазначено особливостей їх застосування в мультитермінальних системах. У [2, 3] розглянуто алгоритм обробки запитів у мультитермінальній системі, проте не висвітлено можливості його застосування при роботі системи в режимі реального часу.

Постановка задачі. Дослідити алгоритм ефективного використання ресурсів мультитермінальної системи, яка обслуговує запити в режимі реального часу.

Основна частина. Сучасні системи орієнтовані переважно на максимальну швидкодію. Її легко досягти в однопрограмній однокористувацькій системі, адже в цьому випадку немає потреби витрачати час на переключення контексту користувача чи задачі. У багатопрограмних режимах роботи системи таке перемикання контексту потрібне, і чим менший період таких переключень, тим більші часові затрати. Якщо багатопрограмна система є багатокористувацькою, то додатково слід перемикати ще й контекст користувача. У таких режимах роботи максимальну швидкодію можуть забезпечити циклічне планування та планування з пріоритетами. Проте це стосується лише випадку, коли виключена можливість простоювання через неактивних користувачів або неактивні задачі. У будь-якому випадку такі системи не гарантують вчасного виконання запитів. Так, можуть траплятися випадки, коли система обслуговує запит, який можна виконати і через годину, тоді як в черзі очікує інший запит, який необхідно виконати протягом 100 мкс. Виконання запитів в режимі суворого нагляду за критичними термінами їх виконання проводять системи, які працюють у режимі реального часу.

Для збільшення ефективності одного робочого місця можна зменшити часові затрати на виконання поставлених перед користувачем завдань. Час, витрачений користувачем на роботу, становить:

$$T = T_L + T_W + T_E, \quad (1)$$

де T_L – час завантаження терміналу (час з моменту приходу користувача на робоче місце до кінця завантаження операційної системи); T_W – час роботи користувача; T_E – час від моменту подання сигналу вимкнення терміналу до його вимкнення.

Оскільки $T_W >> T_L + T_E$, то

$$T \approx T_W. \quad (2)$$

У загальному випадку час роботи залежить від кількості задач, часу їх виконання безпосередньо на комп'ютері, продуктивності роботи користувача та рівня розпаралелення їх виконання:

$$T_W = f(P_U, K_P, t_1, ..., t_i), \quad (3)$$

де P_U – продуктивність користувача; K_P – коефіцієнт розпаралелення виконання задач на комп'ютері; $t_1, ..., t_i$ – час виконання задач.

В кожен момент часу на робочому місці можливий один з таких режимів роботи:

1. Користувач не задіює комп'ютер, комп'ютер нічого не виконує.
2. Користувач не задіює комп'ютер, останній виконує одне або більше завдань.
3. Користувач робить запит, в момент надходження якого комп'ютер нічого не виконував.
4. Користувач робить запит, в момент надходження якого комп'ютер виконував одне або більше завдань.

Час роботи терміналу в першому режимі залежить від продуктивності користувача – чим вона більша, тим меншим буде цей час.

Коефіцієнт розпаралелення можна визначити за такою формулою:

$$K_P = \frac{T_a}{T_s}, \quad (4)$$

де T_a – фактичний час виконання задач на комп'ютері; T_s – загальний час виконання задач за умови їх послідовного виконання.

Час виконання кожної задачі можна подати у вигляді множини квантів часу:

$$A=\{dt_n, \ldots, dt_m\}, \quad (5)$$

де dt_n – перший квант часу для задачі; dt_m – останній квант часу для задачі.

Тоді фактичний час виконання можна визначити як об'єднання множин, кожна з яких представляє одну задачу:

$$Tф = A1 \cap A2 \cap \ldots \cap An. \quad (6)$$

Основна мета даного алгоритму – мінімізувати втрати через невчасне виконання запитів сервером.

Нехай є M процесорів $P_1, \ldots, P_M$ і N задач в черзі.

Якщо $N=0$, то обробка завершена, інакше вибираються запити реального часу, які відсортовуються за резервним часом. Якщо таких немає, то виконується запит найбільш пріоритетного користувача, який після цього видаляється з черги.

Для кожного запиту відбувається перевірка на можливість його вчасного виконання, якщо він виконається за порядком в черзі:

$$T_m^L \geq \sum_{i=1}^{m} T_i^P, \quad (7)$$

де T_m^L – час, який залишився для завершення обробки запиту R_m; T_i^P – час, необхідний на обробку запиту R_i.

Якщо запит R_i не встигає виконатись вчасно за таких умов, то перевіряється можливість його вчасного виконання за умови присвоєння йому першого номеру в черзі. Якщо це можливо, то шукаються вільні процесори і, якщо такі є, запит ставиться першим в їхню чергу запитів. Якщо

таких немає, то обчислюють суму штрафів за невчасне виконання запитів для двох черг – звичайної S_1 та зміненої S_2, в якій цей запит переставлений на перше місце.

$$S = \sum_{i=1}^{N} F_{R_i}\ , \tag{8}$$

де F_{R_i} – штраф за невчасне виконання запиту R_i; N – кількість запитів.

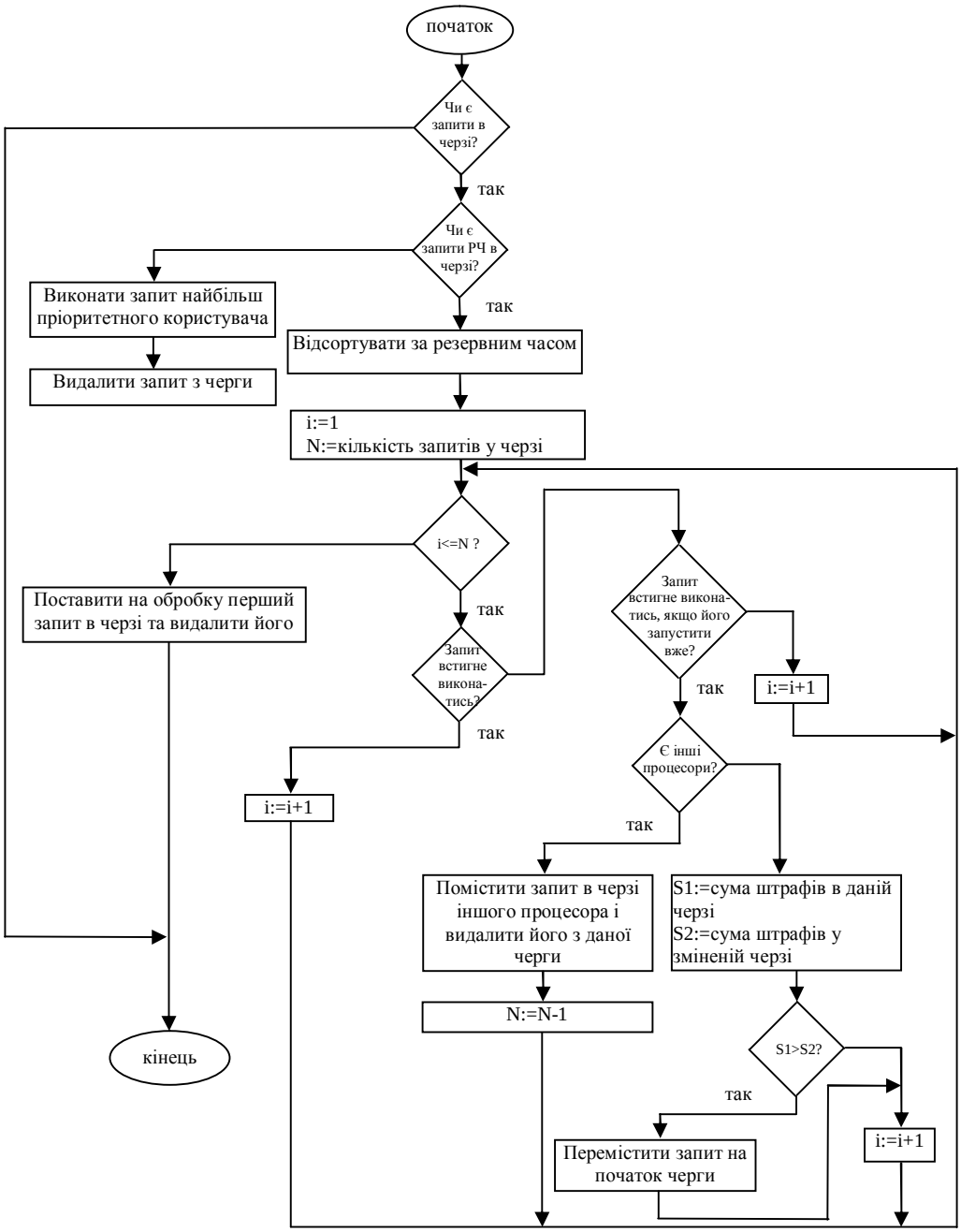

Рис. 1. Граф-схема алгоритму роботи в режимі реального часу

Обробка продовжується для черги, де менший штраф.

Коли заявки надходять з періодом T_i, а на їх виконання потрібен час t_i, то для роботи сервера без перенасичення повинна виконуватися умова:

$$\sum_{i=1}^{N} \frac{t_i}{T_i} \le 1, \tag{9}$$

де N – кількість клієнтів.

Враховуючи, що час обробки заявки залежить від продуктивності сервера:

$$t_i = \frac{n_i}{P}, \tag{10}$$

де n_i – кількість інструкцій, які потрібно виконати в заявці; P – продуктивність сервера

умова набуде такого вигляду:

$$\sum_{i=1}^{N} \frac{n_i}{PT_i} \le 1. \tag{11}$$

Протягом дня розподіл навантаження на сервер є нормальним:

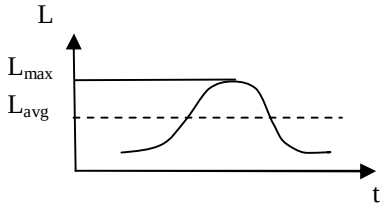

Рис. 2. Графік залежності навантаження на сервер від години дня

де L_{max} – пікове навантаження; L_{avg} – середнє навантаження.

Як правило, $L_{max} = 2L_{avg}$

Нехай максимальна продуктивність сервера, яка може знадобитись одному клієнту, p_{max} . Тоді для гарантування вчасної обробки запитів від усіх клієнтів продуктивність сервера має становити:

$$P = Np_{max}. \tag{12}$$

При цьому:

$$L_{max} = Np_{max}. \tag{13}$$

Проте в цьому випадку при навантаженнях, менших за максимальне, ресурси сервера не будуть використовуватись на 100 %. Тому, коли йдеться не про роботу в реальному часі, а про обробку всіх заявок протягом деякого великого періоду, то вигідніше використовувати сервер з меншою продуктивністю, такою, яка б давала змогу обробляти заявки при середньому наванженні. У загальному випадку – це:

$$P = \frac{Np_{max}}{2}. \tag{14}$$

Якщо є потреба гарантувати роботу в реальному часі певній кількості клієнтів, то необхідну продуктивність сервера можна визначити за формулою:

$$P = \frac{kp_{max}}{2} + (1-k)p_{max}, \tag{15}$$

де k – частка клієнтів, яким треба забезпечити роботу в реальному часі.

Висновки. Розглянуто алгоритм покращеної обробки запитів користувачів у мультитермінальній системі. Його застосування дає змогу ефективно використовувати ресурси мультитермінальної системи та зменшити втрати через невчасне виконання запитів. Це особливо важливо при пікових навантаженнях, коли не всі користувачі отримують відповіді від системи у відведені для цього терміни.

1. Бочаров П.П., Печинкин А.В. Теория массового обслуживания: Учебник. – М: Изд-во РУДН, 1995. – 529 с. 2. Парамуд Я.С., Іващук Т.С. Алгоритм обслуговування запитів пристроїв у мультитермінальній системі // Вісн. Нац. ун-ту „Львівська політехніка" Комп'ютерні системи та мережі. – 2009. – № 658:– С. 116–120. 3. Парамуд Я.С., Іващук Т.С. Структурні рішення під час побудови багатокористувацьких мультитермінальних систем // Вісн. Нац. ун-ту "Львівська політехніка" Комп'ютерні системи та мережі. – 2008. – № 630:– С. 92–96.