УДК 681.3

АНАЛІЗ АЛГОРИТМІВ ВЗАЄМНИХ ВИКЛЮЧЕНЬ КРИТИЧНИХ ІНТЕРВАЛІВ ПРОЦЕСІВ У РОЗПОДІЛЕНИХ СИСТЕМАХ

Бараненко Р.В., Шаганян С.М., Дячук М.В.

Постановка проблеми. Необхідність в усе більш і більш продуктивних системах виникла з першого дня винаходу комп'ютера. На початку це досягалося в результаті еволюції технологій виробництва процесорів. Але також розвивалися тенденції до використання декількох процесорів в одній обчислювальній системі для підвищення продуктивності, розроблялися мультизадачні операційні системи, системи з розподіленою пам'яттю, що підняли ряд проблем, що вимагають швидкого й ефективного рішення [1].

Основною є проблема синхронізації процесів, рішення якої дозволить підвищити продуктивність роботи багатозадачних операційних й багатопроцесорних обчислювальних систем.

Аналіз останніх досліджень. Синхронізація процесів має на увазі для кожного процесу виключення можливості одночасного з ним звертання інших процесів до розподілених даних – ***взаємовиключення.*** Будь-яка спроба взаємного виключення повинна спиратися на якийсь фундаментальний механізм виключень апаратного забезпечення. Найбільш загальним механізмом може служити обмеження, відповідно до якого до деякої комірки пам'яті у визначений момент часу може здійснювати звертання тільки один процес [1].

Під процесом розуміється деяка послідовність дій, що складають деяке обчислення, що характеризується [1,2]:

– зіставленої йому програмою/підпрограмою, тобто впорядкованою послідовністю операцій, що реалізують дії, що повинні здійснюватися процесом;

– вмістом відповідної йому пам'яті, тобто безліччю даних, якими цей процес може маніпулювати;

– дескриптором процесу, тобто сукупністю відомостей, що визначають стан ресурсів, наданих процесові.

Коли процес звертається до розподілених даних, мається на увазі, що він знаходиться у своєму критичному інтервалі [1,2]. Для синхронізації таких процесів використовуються спеціальні захищені глобальні змінні.

В даний час існуючі алгоритми [3-9] успішно вирішують проблему взаємовиключення критичних інтервалів процесів, але володіють при усіх своїх достоїнствах рядом недоліків, тому що використовують для рішення проблеми досить складні способи, коректність яких важко довести; більшість з них складні для програмної реалізації і/або не забезпечують необхідної швидкості взаємодії процесів і ресурсів в ЕОМ.

Мета статті. Метою роботи є проведення порівняльного аналізу алгоритмів взаємного виключення критичних інтервалів процесів й визначення їхніх переваг і недоліків.

Основний матеріал. Застосування глобальних змінних і семафорів вирішує основну проблему синхронізації процесів, що складається у взаємному виключенні критичних інтервалів процесів. У загальному випадку її рішення повинне задовольняти вимогам [10]:

– у будь-який момент часу тільки один процес може знаходитися

 ISBN 7-776-8361-7

всередині критичного інтервалу;

– якщо жоден процес не знаходиться в критичному інтервалі, то будь-який процес, що бажає ввійти до критичного інтервалу, повинний одержати дозвіл без будь-якої затримки;

– жоден процес не повинний нескінченно довго чекати дозволу на вхід до критичного інтервалу (якщо жоден процес не буде знаходитися всередині критичного інтервалу нескінченно);

– не повинно існувати ніяких припущень щодо швидкості процесорів.

Існують різні алгоритми, що можуть бути використані для рішення поставленої задачі в розподілених обчислювальних системах [3-9].

Найбільш простим і найбільш ефективним є ***централізований алгоритм***, заснований на застосуванні тих же методів, що використовуються в однопроцесорних системах. Цей алгоритм гарантує взаємне виключення, але внаслідок своєї централізованої природи має низьку відмовостійкість. При його реалізації один із процесів вибирається як координатор. Коли який-небудь інший процес хоче ввійти до критичного інтервалу, він посилає повідомлення з запитом до координатора, оповіщаючи його про те, до якого критичного інтервалу він хоче ввійти, і чекає від координатора дозволу. Якщо в цей момент жоден із процесів не знаходиться в критичному інтервалі, то координатор посилає дозвіл.

Якщо ж деякий процес уже виконує критичний інтервал, зв'язаний з даним ресурсом, то ніяка відповідь не посилається; запитуючий процес ставиться до черги, і після звільнення критичного інтервалу йому відправляється відповідь-дозвіл.

Розподілений алгоритм діє інакше. Коли процес хоче ввійти до критичного інтервалу, він формує повідомлення, що містить ім'я потрібного йому критичного інтервалу, номер процесу і поточне значення часу. Потім він посилає це повідомлення всім іншим процесам. Передбачається, що передача кожного повідомлення супроводжується підтвердженням. Коли процес одержує повідомлення такого роду, його дії залежать від того, у якому стані стосовно зазначеного в повідомленні критичного інтервалу він знаходиться. Мають місце три ситуації:

1) Якщо одержувач не знаходиться і не збирається входити до критичного інтервалу в даний момент, то він відсилає назад процесові-відправникові повідомлення з дозволом.

2) Якщо одержувач уже знаходиться в критичному інтервалі, то він не відправляє ніякої відповіді, а ставить запит до черги.

3) Якщо одержувач хоче ввійти до критичного інтервалу, але ще не зробив цього, то він порівнює тимчасову оцінку повідомлення, що надійшло, зі значенням часу, що утримується в його власному повідомленні, розісланому всім іншим процесам. Якщо час у повідомленні, що надійшло до нього, менше, тобто його власний запит виник пізніше, то він посилає повідомлення-дозвіл. У зворотному випадку він не посилає нічого і ставить повідомлення-запит, що надійшло до нього, у чергу.

Процес може ввійти до критичного інтервалу тільки в тому випадку, якщо він одержав відповідні повідомлення-дозволи від всіх інших процесів. Коли процес залишає критичний інтервал, він посилає дозвіл усім процесам зі своєї черги і виключає їх з черги.

Зовсім інший підхід до досягнення взаємного виключення в розподілених системах використовує *алгоритм Token Ring*. Усі процеси системи утворюють логічне кільце, тобто кожен процес знає номер своєї позиції в кільці, а також номер найближчого до нього наступного процесу. Коли кільце ініціалізується, процесові *0* передається токен – спеціальна послідовність службової інформації. Токен циркулює по кільцю. Він переходить від процесу n до процесу $n+1$ шляхом передачі повідомлення за типом «крапка-крапка». Коли процес одержує токен від свого сусіда,

він аналізує, чи не потрібно йому самому ввійти до критичного інтервалу. Якщо так, то процес входить до критичного інтервалу. Після того, як процес вийде з критичного інтервалу, він передає токен далі по кільцю. Якщо ж процес, що прийняв токен від свого сусіда, не зацікавлений у входженні до критичного інтервалу, то він відразу відправляє токен у кільце. Отже, якщо один із процесів не бажає входити до критичного інтервалу, то в цьому випадку токен просто циркулює по кільцю з високою швидкістю.

З огляду на переваги й недоліки всіх трьох алгоритмів, можна зробити висновок, що:

1) *Централізований алгоритм* є найбільш простим і найбільш ефективним. При його використанні потрібно тільки три повідомлення для того, щоб процес ввійшов і залишив критичний інтервал: запит, повідомлення-дозвіл для входу і повідомлення про звільнення ресурсу при виході.

2) При використанні *розподіленого алгоритму* для одного використання критичного інтервалу потрібно послати $(n-1)$ повідомлень-запитів (де n - число процесів) - по одному на кожен процес і одержати $(n-1)$ повідомлень-дозволів, тобто усього необхідно $2(n-1)$ повідомлень.

3) В алгоритмі *Token Ring* число повідомлень змінно: від 1 у випадку, якщо кожен процес входив до критичного інтервалу, до нескінченно великого числа, при циркуляції токена по кільцю, у якому жоден процес не входив до критичного інтервалу.

На жаль усі ці три алгоритми погано захищені від відмов. У першому випадку до краху приводить відмова координатора, у другому - відмова будь-якого процесу, а в третьому - втрата токена або відмова процесу.

Висновки. Авторами розглянута проблема синхронізації процесів у розподілених системах, проведений порівняльний аналіз алгоритмів взаємного виключення критичних інтервалів процесів, у результаті якого визначені їхні основні переваги й недоліки, на підставі чого можна аргументувати використання конкретного алгоритму взаємного виключення для підвищення продуктивності роботи багатозадачних операційних й багатопроцесорних систем.

Використання конкретних алгоритмів взаємного виключення критичних інтервалів процесів приведено в працях авторів [1, 10-12].

The problem of synchronization of processes in the distributed systems is considered, the comparative analysis of algorithms of mutual exception of critical intervals of processes is carried out, as a result of which their basic advantages and lacks are determined.

1. С.Н. Шаганян, Р.В. Бараненко Реализация взаимных исключений критических интервалов как одного из видов синхронизации доступа процессов к ресурсам в ЭВМ // Автоматика. Автоматизация. Электротехнические комплексы и системы – Херсон: ХГТУ, 2003, №2(12), С.70-73.

2. Фритч В. Применение микропроцессоров в системах управления: Пер. с нем. – М.: Мир, 1984. - 464 с., ил.

3. Столингс Вильям. Структурная организация и архитектура компьютерных систем, 5-е издание.: пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 896 с.: ил. – Парал. тит. англ.

4. Шоу А. Логическое проектирование операционных систем /Пер. с англ. В.В. Макарова и В.Д. Никитина. – М.: Мир, 1981. – 360 с.

 ISBN 7-776-8361-7

5. Цикритзис Д., Бернстайн Ф. Операционные системы /Пер. с англ. В.Л. Ушковой и Н.Б. Фейгельсон. – М.: Мир, 1977. – 336 с.

6. Системное программное обеспечение /А.В. Гордеев, А.Ю. Молчанов. – СПб.: Питер, 2001. – 736 с.: ил.

7. Peterson G. Myths About the Mutual Exclusion Problem. – Information Processing Letters, June 1981.

8. Hofri M. Proof of a Mutual Exclusion Algorithm. – Operating System Review, January 1990.

9. Столингс Вильям. Операционные системы, 4-е издание.: пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 848 с.: ил. – Парал. тит. англ.

10. С.Н. Шаганян, Р.В. Бараненко Принцип синхронизации доступа к данным в многопроцессорных ЭВМ с общей памятью // Вестник ХГТУ – Херсон: ХГТУ, 2003, №2 (18), С. 289-291.

11. Свідоцтво про реєстрацію авторського права на твір №8645, “Комп′ютерна програма “Автоматизована система прийняття рішень про розподілення інформації за індивідуальними параметрами об′єктів в системах з обхідом тупиків “MedIS”. Автори: Р.В. Бараненко, Т.А. Ілюк, М.В. Пилипенко, Ю.Ю. Синицький, С.Ю. Синицький. Опубл. 23.10.2003.

12. С.Н. Шаганян, Р.В. Бараненко Выполнение запросов о выделении ресурсов в системах с обходами тупиков // Вестник ХГТУ – Херсон: ХГТУ, 2004, №19, С. 63-65.