ДОСЛІДЖЕННЯ МЕТОДІВ ПОБУДОВИ ТРАНЗИКЦІЙНОЇ ПАМ'ЯТІ З АПАРАТНОЇ ПІДТРИМКОЮ І БЕЗ НЕЇ

Стрименешенко О.С.., Шалімов О.А.
Науковий керівник – д. т. н., проф. Іванов В.Г.
Харьковскій національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Системотехники, тел. +38(057)702-10-06)

This topic is very relevant, since lately more there is a transition from sequential to parallel computing, and existing development tools for parallel programs are complex and not effective enough. With development of multi-core processors has created a need for alternative solutions that provide a simple and understandable way to provide access to shared memory. [2]

Транзакційна пам'ять - це зручний спосіб контролю доступу до спільних даних, який пропонує ряд переваг: більш ефективне використання процесорів ядер, оскільки транзакції можуть виконуватися паралельно в різних потоках управління; можливість використання транзакційного коду всередині іншого коду транзакцій, а також інших паралельних механізмів програмування більш простої та зрозумілішої моделі програмного забезпечення. [1]

Транзакційна пам'ять забезпечує механізм транзакцій, що використовується вже багато років в системах баз даних, для потоків управління, виконуваних в загальному адресному просторі. Системи транзакційної пам'яті дозволяють заключити частину коду в транзакцію, яка буде виконуватися атомарно і в ізоляції по відношенню до інших, паралельно що виконуються операціями.

Транзакційна пам'ять може бути реалізована повністю програмно або з використанням спеціальної апаратної підтримки. Крім того, до цих пір, залишається багато питань, яку одиницю даних вибрати для відстеження конфліктів доступу - об'єкти або слова пам'яті; в який момент часу виявляти конфлікти - при доступі до пам'яті або тільки при фіксації транзакції і деякі інші. [3]

У роботі були досліджені основні методи побудови систем транзакційної пам'яті. Були виявлені переваги і недоліки різних підходів, простежено взаємозв'язки між ними для досягнення гарної продуктивності.

В ході дослідження були отримані експериментальні дані, наочно демонструючи порівняльну продуктивність різних підходів, що використовуються при реалізації систем транзакційної пам'яті. Було використано СУБД MySQl. В ній одне з'єднання з БД - одна транзакція. Нова транзакція в межах одного з'єднання може початися тільки після завершення попередньої.

В разі SQL помилки, транзакція сама по собі не відкотиться. Зазвичай помилки обробляються вже за допомогою sql wrapper'ів в самому додатку. Якщо ви захочете відкочувати зміни в разі помилки прямо в MySQL, можна створити спеціальну процедуру і вже в ній виконувати ROLLBACK(відкотити транзакцію) в обробнику.

Результати тестування показали, що продуктивність тієї чи іншої системи транзакційної пам'яті залежить від конкретного додатка, в якому вона використовується. Було запропоновано об'єднати різні підходи і дозволяти програмістам явно вказувати, яким чином повинна працювати система в конкретній ситуації. Це дозволить зробити транзакційну пам'ять більш універсальною.

Також на основі отриманих даних були зроблені висновки про ефективність деяких методів побудови транзакційної пам'яті. Найбільш перспективним рішенням за результатами дослідження є програмна транзакційна пам'ять з відкладеними змінами, раннім виявленням конфліктів і використанням апаратного прискорення для скорочення накладних витрат, необхідних для забезпечення механізму транзакцій.

Перелік посилань:

1. Larus, J., Kozyrakis, C. Transactional memory // Commun. ACM 51, 7 (Jul.2008), P.80-88 [PDF] (http://doi.acm.org/10.1145/1364782.1364800).

2. Harris, T., Marlow, S., Jones, S. P., and Herlihy, M. Composable memory transactions. Commun. ACM 51, 8 (Aug. 2008), P. 91-100 [PDF] (http://doi.acm.org/10.1145/1378704.1378725).

3. Saha, B., Adl-Tabatabai, A., and Jacobson, Q. Architectural Support for Software Transactional Memory // Proceedings of the 39th Annual IEEE/ACM international Symposium on Microarchitecture (December 09 - 13, 2006). IEEE Computer Society, Washington, DC, P. 185-196 [PDF] (http://dx.doi.org/10.1109/MICRO.2006.9)