ƯU ĐIỂM CỦA LẬP TRÌNH MULTI-CORE

Russell Hitchcock

Nếu là bạn đọc từ lâu của Quantrimang.com thì chắc hẳn các bạn đã biết tới bài giới thiệu về CPU đa lõi. Đây là bài viết mà trong đó chúng tôi đã giới thiệu một chút về ưu thế của các phần mềm trong việc sử dụng hệ thống đa lõi. Trong bài này chúng tôi sẽ giới thiệu sâu hơn về thế giới multithread (tạm dịch là thế giới đa luồng) và đi vào xem xét một số cách có thể khắc phục các vấn đề khó khăn trong việc phát triển ứng dụng chạy trên nền tảng đa lõi.

J2EE

Công nghệ ứng dụng là thứ mà có lẽ một số bạn có thể đã biết đó là J2EE, J2EE là một môi trường được sử dụng để xây dựng các ứng dụng mức doanh nghiệp. Các ứng dụng được xây dựng trên công nghệ J2EE có thể tận dụng nhiều lợi thế trong việc sử dụng các CPU đa lõi.

Việc sử dụng J2EE cho các ứng dụng doanh nghiệp sẽ cho phép các chuyên gia phát triển viết mã ứng dụng mà không cần đến các kiến thức chi tiết sâu về thread; đây quả thực là sức mạnh thực sự nằm phía sau J2EE. Các ứng dụng J2EE được chuyển đổi thành multithread bằng cách thay đổi các thiết lập có trong máy chủ ứng dụng; việc thực hiện multithread được quản lý bởi một mục (container). Vậy cách thức thực hiện như thế nào. Về cơ bản, sự logic doanh nghiệp có thể được viết mã và sau đó được phát triển vào máy chủ trong một mục logic. Sau đó các thiết lập cũng có thể phân định số lượng bao nhiêu các trường hợp phần logic doanh nghiệp có thể được chạy. Mục cũng có thể phân định các kết nối đến các tài nguyên chẳng hạn như cơ sở dữ liệu có thể được sử dụng một lần. Các thuộc tính được quản lý bởi các mục này hoàn toàn tách biệt với việc multithread đối với mã ứng dụng. Đây là một dấu hiệu tốt vì nó cho phép việc multithread có thể được điều chỉnh dựa trên phần cứng của ứng dụng đang chạy mà không cần thay đổi mã ứng dụng. Điều này thích hợp với phần cứng khối doanh nghiệp; các máy tính có thể được bổ sung vào cụm các máy chủ (server cluster), hoặc có lẽ các bộ vi xử lý có thể được bổ sung hoặc được tách bớt từ các máy tính đang tồn tại trên cụm, mỗi một sự thay đổi này đều ảnh hưởng đến khả năng multi-thread của các ứng dụng dang chạy trên phần cứng.

OpenMP

Việc nhớ rằng multithread logic hoàn toàn tách biệt với logic doanh nghiệp sẽ là một mục tiêu chính cho các chuyên gia phát triển phần mềm viết các ứng dụng song song. Có nhiều lý do cho vấn đề này; dễ dàng cho việc phát triển, dễ dàng cho việc gỡ rối và dễ dàng thay đổi ứng dụng. Khi một chuyên gia phát triển phần mềm đang phát triển một ứng dụng trong C/C++ hoặc FORTRAN thì một thực thi phổ biến được sử dụng cho quá trình này đó là OpenMP. OpenMP là một API được sử dụng để viết mã ứng dụng đa luồng một cách hiệu quả và thiết thực.

Về bản chất, OpenMP được sử dụng bởi việc nhóm một tập các chỉ lệnh trong mã dưới dạng các comment hoặc các annotation (chú thích). Mã được viết và sau đó các chú thích được bổ sung vào các vị trí thích hợp sau đó. Khi mã đã được biên dịch với bộ biên dịch OpenMP thích hợp, các chú thích sẽ được đọc và mã được biên dịch theo cách như vậy để tạo sử dụng cho các thread trực tiếp bởi các chú thích.

Phương pháp lập trình song song này rất có nhiều ưu điểm. Do một chương trình được viết mã để có thể chạy tuần tự và các chú thích chỉ đơn thuần là các comment của mã khi đó nếu mã được biên dịch trong một bộ biên dịch thông thường thì bộ biên dịch này sẽ bỏ qua các chú thích

OpenMP. Mã này sau đó sẽ được biên dịch với bộ biên dịch OpenMP và chạy đa luồng một cách song song. Điều này có nghĩa rằng các chuyên gia phát triển sẽ không phải thay đổi mã nếu chương trình được chạy trong hai kiến trúc phần cứng khác nhau: một cho đa luồng và một không cho đa luồng.
Một ưu điểm khác của OpenMP là phần mã đó có thể được chú thích gia tăng với nhiều mã. Điều đó cho phép việc kiểm tra dễ dàng hơn trong việc bảo đảm đúng chức năng mã. Vấn đề này cũng rất có ý nghĩa vì các chuyên gia phát triển có thể xử lý đa luồng nhiều phần mã để sau đó sử dụng kết quả trong một thực thi khác; tuy nhiên điều này rất khó để có thể kiểm tra bởi một bộ biên dịch vì vậy cần phải được kiểm tra thông qua việc kiểm thử.

Hình 1: Ví dụ về một luồng ứng dụng sử dụng multithread
Những nguy hiểm của Multithread
Một trong những lỗi hay gặp nhất được tạo ra bởi các chuyên gia phát triển khi thiết kế đa luồng các ứng dụng của họ là việc xử lý các vòng lặp. Cho ví dụ, nếu một ai đó muốn lặp lại toàn bộ một mảng các số nguyên và chỉ bổ sung một cách đơn giản vào mỗi giá trị đó thì điều này có thể được xử lý đa luồng để thực hiện mỗi hành động lặp trong thread của chính nó tại cùng một thời điểm. OpenMP sẽ thực hiện điều này (trong C/C++) bởi chú thích #pragma omp parallel for. Tuy nhiên nếu ai đó muốn lặp lại một mảng tương tự đó nhưng thay đổi mỗi giá trị bằng cách thêm vào giá trị của thành phần trước thì trong trường hợp này sẽ gặp phải một lỗi tróng xử lý đa luồng vòng lặp này; nếu tất cả các lần lặp của vòng lặp xuất hiện một cách đồng thời thì bạn không thể bảo đảm rằng thành phần trước đó đã được thay đổi để có thể chứa đựng giá trị mới. Việc chạy một vòng lặp như vậy trong quá trình xử lý đa luồng sẽ tạo ra một mảng số nguyên khác.
Một lỗi khác nữa cũng hay xảy ra đó là với các biến được sử dụng bởi quá trình xử lý đa luồng đồng thời. OpenMP về cơ bản được sử dụng cho các chương trình chạy trong chế độ bộ nhớ chia sẻ và chính vì vậy yêu cầu các chuyên gia phát triển phải bảo đảm rằng các luồng này không làm thay đổi các biến được sử dụng bởi các luồng khác. Tuy nhiên trong trường hợp này, OpenMP lại có các chú thích được hỗ trợ để bảo đảm cho điều đó không xảy ra bên trong một khối mã xử lý đa luồng nào đó. Tuy nhiên các chuyên gia vẫn phải tìm hiểu về điều này để sử dụng đúng các chú thích ở nơi cần thiết.
Hiệu suất
Khi được thực hiện một cách cẩn thận, việc xử lý các ứng dụng đa luồng có thể làm tăng hiệu suất một cách đáng kể. Rõ ràng để thực hiện điều này thì bạn phải chạy trên các phần cứng có hỗ trợ cho việc xử lý đa luồng. Tuy nhiên khi thực hiện với phần cứng thích hợp thì cũng vẫn có một số hạn chế nhất định đối với việc xử lý này, chẳng hạn như có các khối mã không thể xử lý đa luồng. Nhiều phần mã không thể được thực thi đa luồng sẽ làm tăng gian thực thi của các phần mã đó và từ đó sẽ gây ảnh hưởng đến hiệu suất của toàn bộ chương trình.
Mặc dù vậy nhưng có rất nhiều ưu điểm mang lại cho hiệu suất thực thi để thực hiện chạy một ứng dụng đa luồng trên các CPU đa lõi đó là có thêm nhiều bộ nhớ cho ứng dụng. Cho ví dụ, bộ vi xử lý Tile64 của Tilera đã được giới thiệu đến trong bài về CPU đa lõi có chứa một cache logic L3. Trong nhiều trường hợp, cách thức xử lý này sẽ tăng được hiệu suất sử dụng bộ nhớ cache một cách đáng kể, từ đó giảm được thời gian bộ nhớ cần đến trong việc đọc và làm tăng hiệu suất thực thi.