TÍNH TOÁN SONG SONG VÀ LẬP TRÌNH OPENMP TRONG BỐI CẢNH KIẾN TRÚC ĐA LÕI VÀ TÍNH TOÁN DỊ THỂ

Vũ Thị Thu Hiền
Khoa Công nghệ Thông tin, Trường Đại học Công nghệ Kỹ thuật Vinh
Email: thuhienktv@gmail.com

***Tóm tắt**: Sự phát triển của trí tuệ nhân tạo, phân tích dữ liệu lớn, mô phỏng khoa học và các ứng dụng tính toán hiệu năng cao làm gia tăng nhu cầu khai thác đồng thời nhiều đơn vị xử lý. Tính toán song song vì vậy trở thành một thành phần quan trọng của kiến trúc máy tính hiện đại, từ bộ xử lý đa lõi đến hệ thống gồm CPU, GPU và các bộ tăng tốc. Bài viết trình bày khái quát về tính toán song song, các dạng kiến trúc theo dòng lệnh và dòng dữ liệu, sự khác biệt giữa thực thi đồng bộ và không đồng bộ, đồng thời tập trung vào mô hình lập trình OpenMP. Bên cạnh các cấu trúc truyền thống như vùng song song, chia sẻ công việc và đồng bộ luồng, bài viết cập nhật xu hướng OpenMP trong hỗ trợ task, quản lý dữ liệu và offloading trên các thiết bị tăng tốc. Qua đó, bài viết làm rõ vai trò của OpenMP như một mô hình lập trình tương đối đơn giản, khả chuyển và phù hợp để khai thác tài nguyên tính toán đa lõi và hệ thống dị thể.*

***Từ khóa:** Tính toán song song; kiến trúc song song; đa lõi; OpenMP; lập trình đa luồng; tính toán dị thể.*

PARALLEL COMPUTING AND OPENMP PROGRAMMING IN THE CONTEXT OF MULTICORE ARCHITECTURES AND HETEROGENEOUS COMPUTING

***Abstract:** The development of artificial intelligence, big data analytics, scientific simulations, and high-performance computing applications has increased the demand for simultaneously exploiting multiple processing units. Parallel computing has therefore become an essential component of modern computer architectures, ranging from multicore processors to systems integrating CPUs, GPUs, and various accelerators. This paper provides an overview of parallel computing, different parallel architectures based on instruction and data streams, and the distinctions between synchronous and asynchronous execution, with a particular focus on the OpenMP programming model. In addition to traditional constructs such as parallel regions, work-sharing, and thread synchronization, the paper discusses recent developments in OpenMP, including tasking, data management, and offloading to accelerator devices. These developments highlight the role of OpenMP as a relatively simple and portable programming model for effectively exploiting computational resources in multicore and heterogeneous systems.*

***Keywords**: Parallel computing; parallel architectures; multicore; OpenMP; multithreaded programming; heterogeneous computing.*

Nhận bài: 20/09/2026 Phản biện: 23/10/2026 Duyệt đăng: 27/10/2026

I. ĐẶT VẤN ĐỀ

Trong nhiều thập niên, hiệu năng máy tính chủ yếu được cải thiện thông qua việc tăng tốc độ bộ xử lý. Tuy nhiên, những giới hạn về năng lượng, nhiệt lượng và công nghệ chế tạo khiến việc tiếp tục tăng tần số không còn là hướng duy nhất để nâng cao hiệu năng. Thay vào đó, kiến trúc đa lõi, xử lý song song, bộ xử lý vector, GPU và các bộ tăng tốc ngày càng được sử dụng rộng rãi. Tính toán song song cho phép phân chia một bài toán thành nhiều phần có thể thực hiện đồng thời, từ đó rút ngắn thời gian xử lý đối với những bài toán có khối lượng tính toán lớn (Nguyễn Kim Khánh, n.d.; Phạm Văn Cường, n.d.).

Trong thực tế, tính toán song song đã trở thành nền tảng của nhiều lĩnh vực như mô phỏng khoa học, xử lý dữ liệu lớn, trí tuệ nhân tạo, đồ họa, phân tích tài chính và tính toán hiệu năng cao. Tại Việt Nam, nội dung kiến trúc máy tính song song, mô hình lập trình và OpenMP đã được đưa vào chương trình đào tạo ở nhiều cơ sở giáo dục đại học, cho thấy vai trò ngày càng rõ của kiến thức song song trong đào tạo nhân lực công nghệ thông tin (Ngô Văn Thanh, 2009; Trường Đại học Đồng Tháp, n.d.).

Một vấn đề quan trọng là khả năng chuyển một chương trình tuần tự thành chương trình song song không chỉ phụ thuộc vào số lượng bộ xử lý mà còn phụ thuộc vào cách phân chia công việc, tổ chức dữ liệu, chi phí giao tiếp và đồng bộ. Vì vậy, nghiên cứu kiến trúc song song cần được gắn với mô hình lập trình và các cơ chế kiểm soát thực thi. Trong số các mô hình lập trình hiện nay, OpenMP cung cấp phương thức tương đối đơn giản để khai thác bộ nhớ dùng chung trên các hệ thống đa lõi, đồng thời hỗ trợ các hệ thống có thiết bị tăng tốc (OpenMP Architecture Review Board, 2024).

II. KIẾN TRÚC VÀ MÔ HÌNH TÍNH TOÁN SONG SONG

2.1. Tổng quan về tính toán song song

Tính toán song song là phương thức tổ chức trong đó nhiều phép tính được thực hiện đồng thời nhằm giải quyết một bài toán chung. Mức độ song

song có thể được khai thác ở nhiều cấp, từ mức bit, mức lệnh, mức dữ liệu đến mức tác vụ. Trong kiến trúc hiện đại, bộ xử lý đa lõi cho phép nhiều luồng thực thi đồng thời, trong khi GPU và các bộ tăng tốc cung cấp khả năng xử lý song song quy mô lớn đối với những bài toán có tính song song dữ liệu cao (Nguyễn Kim Khánh, n.d.; Phạm Văn Cường, n.d.).

Hiệu quả của chương trình song song không đơn giản tỷ lệ với số bộ xử lý. Khi phân chia bài toán, phần công việc tuần tự, chi phí tạo và quản lý luồng, trao đổi dữ liệu, truy cập bộ nhớ và đồng bộ có thể làm giảm mức tăng tốc. Do đó, đánh giá chương trình song song thường quan tâm đến speedup, efficiency, khả năng mở rộng và mức sử dụng tài nguyên (Nguyễn Kim Khánh, n.d.; Phạm Văn Cường, n.d.).

2.2. Các kiến trúc song song

Một cách phân loại kinh điển là phân loại Flynn dựa trên số dòng lệnh và dòng dữ liệu. SISD tương ứng với thực thi tuần tự; SIMD thực hiện cùng một lệnh trên nhiều phần tử dữ liệu; MISD có nhiều dòng lệnh xử lý cùng một dòng dữ liệu và ít phổ biến; MIMD cho phép nhiều dòng lệnh xử lý trên nhiều dòng dữ liệu và là mô hình phổ biến trong các hệ thống đa lõi, đa bộ xử lý và cụm máy tính (Ngô Văn Thanh, 2009; Phạm Văn Cường, n.d.).

Bên cạnh cách phân loại theo Flynn, kiến trúc song song còn có thể được xem xét theo tổ chức bộ nhớ. Hệ thống bộ nhớ dùng chung cho phép các luồng hoặc bộ xử lý truy cập một không gian địa chỉ chung, thuận lợi cho lập trình đa luồng nhưng đòi hỏi kiểm soát truy cập dữ liệu. Ngược lại, hệ thống bộ nhớ phân tán sử dụng nhiều nút tính toán với bộ nhớ riêng và thường cần cơ chế trao đổi thông điệp. Trong thực tế, các hệ thống hiện đại có thể kết hợp nhiều mô hình, hình thành kiến trúc dị thể gồm CPU, GPU và các bộ tăng tốc (Ngô Văn Thanh, 2009; Phạm Văn Cường, n.d.).

2.3. Đồng bộ và không đồng bộ trong tính toán song song

Trong thực thi đồng bộ, các luồng hoặc tiến trình phải tuân theo những điểm phối hợp xác định trước. Đồng bộ giúp bảo đảm thứ tự hoặc điều kiện cần thiết khi nhiều luồng cùng truy cập dữ liệu, nhưng nếu sử dụng không hợp lý có thể làm phát sinh thời gian chờ và giảm hiệu năng. Các cơ chế như barrier, critical và lock thường được sử dụng để kiểm soát truy cập và phối hợp giữa các luồng (Ngô Văn Thanh, 2009; OpenMP Architecture Review Board, 2024).

Trong thực thi không đồng bộ, các tác vụ có thể được thực hiện độc lập hoặc tiếp tục thực hiện trong khi một công việc khác đang chờ dữ liệu hoặc tài nguyên. Mô hình này đặc biệt có ý nghĩa với các bài toán có sự phụ thuộc không đều giữa các công việc. OpenMP hỗ trợ lập trình task và biểu diễn quan hệ phụ thuộc giữa các task, giúp chương trình khai thác tốt hơn mức song song động (OpenMP Architecture Review Board, 2024).

Xu hướng chuyển từ mô hình chỉ dựa trên các vòng lặp song song sang kết hợp song song dữ liệu, song song tác vụ và thực thi bất đồng bộ phù hợp với các bài toán có cấu trúc phức tạp, dữ liệu động và hệ thống dị thể (OpenMP Architecture Review Board, 2024).

III. LẬP TRÌNH SONG SONG VỚI OPENMP

OpenMP là API lập trình song song dựa trên chỉ thị, thư viện thời gian chạy và biến môi trường, hỗ trợ C, C++ và Fortran, chủ yếu cho các hệ thống bộ nhớ dùng chung. Với mô hình đơn giản và khả chuyển, OpenMP cho phép bổ sung các chỉ thị vào chương trình tuần tự để tổ chức thực thi đa luồng mà không phải trực tiếp quản lý từng luồng (OpenMP Architecture Review Board, 2024).

OpenMP sử dụng mô hình fork-join: khi vào vùng song song, luồng chính tạo nhóm luồng để thực hiện công việc; sau đó các luồng được hợp nhất và chương trình tiếp tục thực thi. Các cấu trúc cơ bản gồm parallel, for, sections, single, task và các cơ chế đồng bộ. Trong đó, parallel for thường được dùng để song song hóa các vòng lặp có các lần lặp tương đối độc lập (OpenMP Architecture Review Board, 2024). Tuy nhiên, song song hóa không phải lúc nào cũng làm chương trình nhanh hơn do chi phí tạo luồng, lập lịch và đồng bộ; hiệu quả còn phụ thuộc vào kích thước bài toán, cân bằng tải, băng thông bộ nhớ và phụ thuộc dữ liệu (Nguyễn Kim Khánh, n.d.; Phạm Văn Cường, n.d.).

Để bảo đảm tính đúng đắn, OpenMP cung cấp các cơ chế critical, atomic, barrier, reduction và ordered. Nếu nhiều luồng đồng thời truy cập và cập nhật dữ liệu dùng chung mà không có cơ chế phù hợp, có thể xảy ra race condition. Trong các bài toán tính tổng hoặc phép toán kết hợp, reduction cho phép mỗi luồng tạo kết quả cục bộ rồi kết hợp ở cuối, qua đó giảm tranh chấp và chi phí đồng bộ so với sử dụng critical (Ngô Văn Thanh, 2009; OpenMP Architecture Review Board, 2024).

OpenMP cũng cho phép xác định phạm vi dữ liệu bằng các thuộc tính shared, private, firstprivate và reduction. Trong đó, shared cho

phép các luồng cùng truy cập một biến; private tạo bản sao riêng cho từng luồng; firstprivate tạo bản sao riêng và khởi tạo từ giá trị ban đầu; reduction tạo các bản sao cục bộ và kết hợp kết quả. Xác định đúng phạm vi dữ liệu giúp hạn chế lỗi cạnh tranh, giảm đồng bộ không cần thiết và nâng cao hiệu năng chương trình (OpenMP Architecture Review Board, 2024).

IV. XU HƯỚNG OPENMP TRONG TÍNH TOÁN HIỆN ĐẠI

OpenMP đang chuyển từ mô hình chủ yếu khai thác CPU đa lõi sang mô hình hỗ trợ các hệ thống dị thể. OpenMP 4.x đã đưa vào khả năng offloading lên thiết bị tăng tốc và các phiên bản mới tiếp tục mở rộng khả năng quản lý dữ liệu, thiết bị và bộ nhớ. OpenMP 6.0 tiếp tục phát triển các khả năng về tasking, phụ thuộc task, truyền dữ liệu và quản lý tài nguyên bộ nhớ (OpenMP Architecture Review Board, 2024).

Điều này có ý nghĩa đặc biệt trong bối cảnh các ứng dụng trí tuệ nhân tạo, mô phỏng và phân tích dữ liệu ngày càng sử dụng GPU hoặc các bộ tăng tốc. Thay vì phải xây dựng toàn bộ chương trình theo một mô hình phần cứng riêng, OpenMP hướng tới việc cung cấp một lớp lập trình có tính khả chuyển, cho phép lập trình viên mô tả mức song song và dữ liệu cần xử lý trên thiết bị đích (OpenMP Architecture Review Board, 2024).

Một xu hướng khác là khai thác task-based parallelism. Với những bài toán mà các công việc được tạo động hoặc có quan hệ phụ thuộc phức tạp, việc mô tả từng task và quan hệ phụ thuộc giúp hệ thống thời gian chạy có thể lập lịch linh hoạt hơn. Đây là hướng phù hợp với các ứng dụng có cấu trúc không đều và yêu cầu khai thác đồng thời nhiều loại tài nguyên (OpenMP Architecture Review Board, 2024).

Do đó, việc học OpenMP hiện nay không nên dừng ở các chỉ thị parallel và for, mà cần hướng đến ba năng lực: phân tích khả năng song song của bài toán, tổ chức dữ liệu và đồng bộ đúng, và khai thác tài nguyên dị thể khi cần thiết. Cách tiếp cận này giúp kiến thức OpenMP gắn với yêu cầu thực tế của tính toán hiệu năng cao và các ứng dụng dữ liệu lớn.

V. KẾT LUẬN

Tính toán song song là một hướng quan trọng của kiến trúc và phát triển phần mềm hiện đại, đặc biệt trong bối cảnh bộ xử lý đa lõi, GPU, điện toán hiệu năng cao và trí tuệ nhân tạo phát triển mạnh. Việc phân loại kiến trúc theo dòng lệnh, dòng dữ liệu và tổ chức bộ nhớ giúp làm rõ nền tảng phần cứng; trong khi các mô hình đồng bộ, không đồng bộ và task-based cung cấp những cách thức khác nhau để khai thác mức song song của bài toán.

OpenMP cung cấp một mô hình lập trình tương đối đơn giản nhưng có khả năng mở rộng, từ lập trình đa luồng trên CPU đến hỗ trợ thiết bị tăng tốc và hệ thống dị thể. Các cơ chế vùng song song, chia sẻ công việc, quản lý dữ liệu và đồng bộ là nền tảng để xây dựng chương trình đúng và hiệu quả. Sự phát triển của OpenMP 6.0 cho thấy lập trình song song đang chuyển mạnh sang tasking, thực thi bất đồng bộ, quản lý bộ nhớ và offloading (OpenMP Architecture Review Board, 2024).

Trong đào tạo công nghệ thông tin, việc kết hợp kiến thức kiến trúc song song với thực hành OpenMP có thể giúp người học chuyển từ tư duy lập trình tuần tự sang tư duy phân chia công việc, quản lý dữ liệu và đánh giá hiệu năng. Đây cũng là cơ sở để tiếp cận các mô hình tính toán hiện đại như GPU computing, heterogeneous computing và high-performance computing.

TÀI LIỆU THAM KHẢO

Ngô Văn Thanh. (2009). Bài giảng tính toán song song. Viện Vật lý, Viện Hàn lâm Khoa học và Công nghệ Việt Nam.

Nguyễn Kim Khánh. (n.d.). Bài giảng lập trình song song. Trường Đại học Bách khoa Hà Nội.

OpenMP Architecture Review Board. (2024). OpenMP application programming interface specification (Version 6.0). https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-6-0.pdf

Phạm Văn Cường. (n.d.). Bài giảng tính toán song song. Khoa Công nghệ thông tin 1, Học viện Công nghệ Bưu chính Viễn thông.

Trường Đại học Đồng Tháp. (n.d.). Đề cương học phần tính toán song song [Đề cương học phần, Chương trình đào tạo Khoa học máy tính].