ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TỐI ƯU PHẦN MỀM NHÚNG TRÊN BỘ XỬ LÝ ĐA NHÂN

Bùi Hữu Phúc

TÓM TẮT LUẬN ÁN TIẾN SĨ

Hà Nội – 2022

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TỐI ƯU PHẦN MỀM NHÚNG TRÊN BỘ XỬ LÝ ĐA NHÂN

Bùi Hữu Phúc

Cán bộ hướng dẫn: PGS. TS. Nguyễn Ngọc Bình
TS. Lê Quang Minh

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật phần mềm

TÓM TẮT LUẬN ÁN TIẾN SĨ

Hà Nội – 2022

MỤC LỤC

MỞ ĐẦU

1. Đặt vấn đề

Với sự phát triển mạnh mẽ của mình, công nghệ phần mềm nhúng ngày càng được nghiên cứu sâu rộng và hoàn thiện đặc biệt là phần mềm nhúng trên môi trường di động. Các thiết bị nhúng thường bị giới hạn về: khả năng xử lý CPU, kích thước bộ nhớ, thời gian sống của pin, vấn đề tiêu thụ năng lượng, vấn đề thời gian thực, v.v. Do đó việc nghiên cứu vấn đề tối ưu phần mềm nhúng có ý nghĩa đặc biệt quan trọng.

Các nghiên cứu đưa ra nhiều khía cạnh nhằm tối ưu hiệu năng của phần mềm, nhưng vẫn còn các hạn chế như: Nguy cơ mất cân bằng nên không đạt được tốc độ xử lý; Khả năng mở rộng của song song tác vụ bị hạn chế do một tác vụ nhất định chỉ có thể được chia thành một tập hợp giới hạn các tác vụ con cho đến khi đạt đến một tác vụ nguyên tử nên nhiều khả năng, chi phí phân phối xử lý cao hơn lợi nhuận thu được từ việc tiếp tục sự song song hóa; Cần có sự cân bằng tải rõ ràng giữa các tác vụ khi các dữ liệu được phân phối không đồng đều trên dữ liệu đầu vào.

Từ ý nghĩa đặc biệt quan trọng, sự cần thiết trong xu hướng phát triển của IoT và Công nghiệp 4.0 là một hướng nghiên cứu còn mới, chúng tôi nhận thấy: "***Tối ưu phần mềm nhúng trên Bộ xử lý đa nhân***" là cần thiết cả về mặt khoa học và thực tiễn.

2. Mục tiêu, đối tượng và phạm vi nghiên cứu

- Cải tiến phương pháp tối ưu trong giai đoạn cài đặt, đặc biệt tập trung vào tối ưu mã nguồn cho các bộ xử lý đa;

- Xây dựng được mô hình tối ưu hướng mã nguồn cho phần mềm nhúng trên bộ xử lý đa nhân.

3. Những đóng góp chính của luận án

- Đề xuất phương pháp *lựa chọn các hàm xử lý chính theo luật Pareto 8/2 và dựa trên hàm điều kiện để thực thi đa luồng nhằm xác định số tác vụ thực thi song song hóa.* Việc xác định các tác vụ chính bằng cách *tần suất gọi hàm và số vòng lặp, số dòng lệnh trong hàm, xác định được điều kiện ràng buộc số tác vụ được song song hóa* để tăng hiệu năng của phần mềm nhúng;

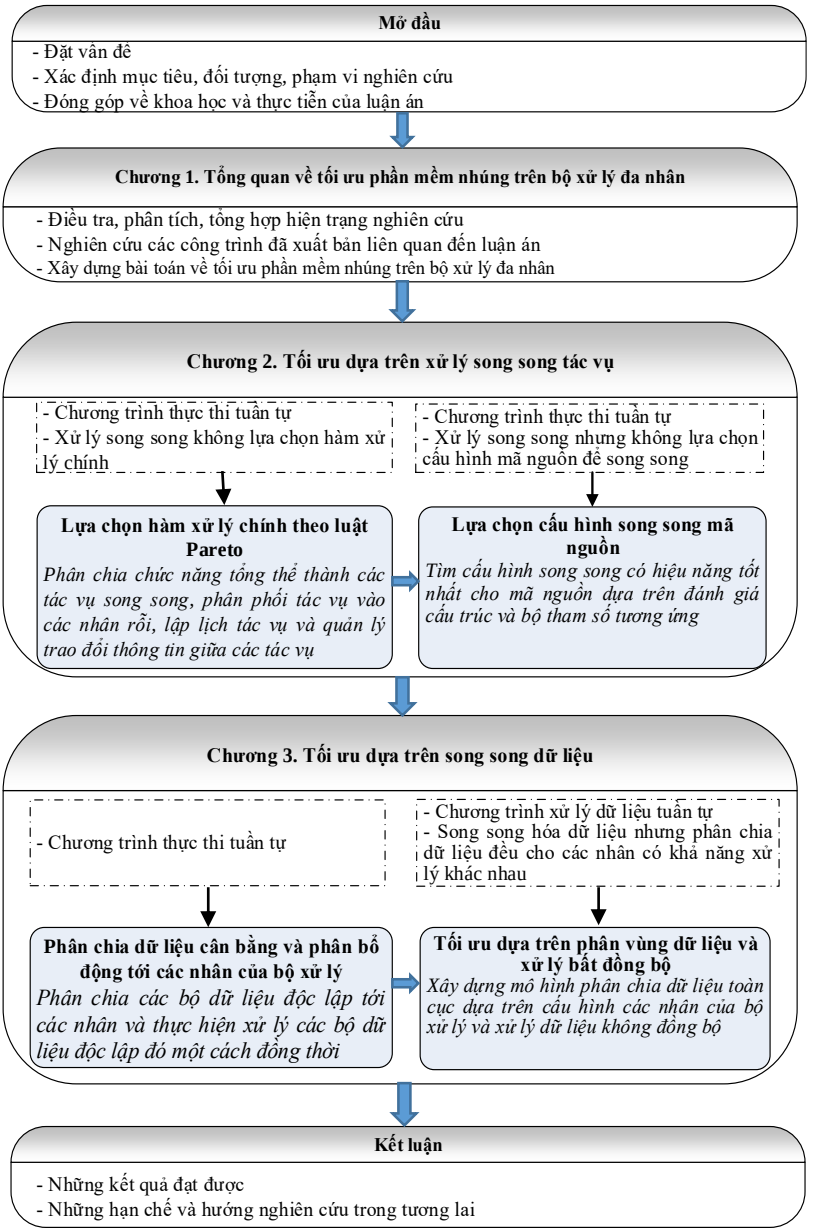
+ Đề xuất phương pháp *tối ưu cấu hình mã nguồn phần mềm nhúng*, dựa trên *tìm cấu hình song song có hiệu năng tốt nhất cho mã nguồn thông qua đánh giá cấu trúc và bộ tham số tương ứng* của mã nguồn sinh ra phần mềm nhúng có hiệu năng tốt nhất trên bộ xử lý đa nhân;

- Đề xuất phương pháp *phân chia dữ liệu cân bằng và phân bổ động nhằm song song hóa dữ liệu để cải tiến hiệu năng phần mềm nhúng* dựa trên hướng tiếp cận *phân chia các bộ dữ liệu độc lập tới các nhân* và thực hiện x*ử lý các bộ dữ liệu độc lập đó một cách đồng thời* nhằm tăng hiệu năng cho phần mềm nhúng trên bộ xử lý đa nhân;

+ Đề xuất phương pháp *phân vùng dữ liệu và xử lý bất đồng bộ* dựa trên việc xây dựng *mô hình phân chia dữ liệu toàn cục theo cấu hình các nhân* của bộ xử lý và *xử lý dữ liệu không đồng bộ giữa các luồng* để cải tiến hiệu năng phần mềm nhúng trên bộ xử lý đa nhân.

4. Bố cục của luận án

Luận án bao gồm mở đầu, ba chương và kết luận. Trong đó, Mở đầu trình bày đặt vấn đề, mục tiêu, đối tượng, phạm vi nghiên cứu và những đóng góp chính của luận án, các Chương còn lại được cấu trúc như Hình 1.



Hình 1. Bố cục của luận án

CHƯƠNG 1. TỔNG QUAN VỀ TỐI ƯU PHẦN MỀM NHÚNG TRÊN BỘ XỬ LÝ ĐA NHÂN

1.1. Một số khái niệm

Các khái niệm cơ sở như: *Phần mềm nhúng; Xử lý song song; Xử lý đồng bộ và bất đồng bộ; Kỹ nghệ đảo ngược* được trình bày để làm rõ các phương pháp luận trong các đề xuất của luận án cũng như trong các nghiên cứu liên quan.

1.2. Một số nghiên cứu về tối ưu phần mềm nhúng trên bộ xử lý đa nhân

Các công trình khoa học ở trong nước và trên thế giới về: *Tối ưu phần mềm nhúng; Xử lý song song trên bộ xử lý đa nhân; Đồng bộ, bất đồng bộ và phân chia dữ liệu; IoT và môi trường phát triển* được trích dẫn và trình bày để tìm ra khoảng trống nghiên cứu cũng như các kiến thức nền tảng để triển khai các hướng tiếp cận và nghiên cứu, thực nghiệm cho các đề xuất của luận án.

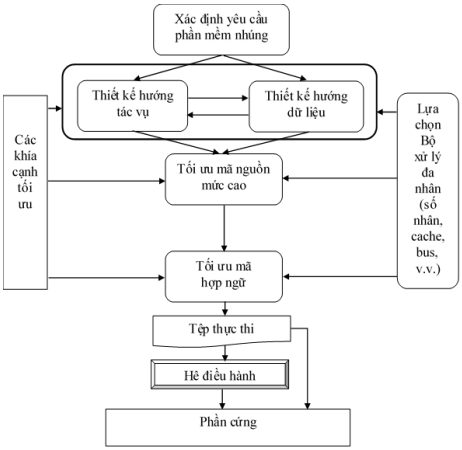
1.3. Bài toán Tối ưu phần mềm nhúng trên bộ xử lý đa nhân

1.3.1. Một số thách thức trong tối ưu phần mềm nhúng trên bộ xử lý đa nhân

Các thách thức đó phải kể đến: *Tính mới trong vấn đề tối ưu trên bộ xử lý đa nhân; Thay đổi tư duy trong lập trình; Điều khiển, tính toán phân chia tác vụ/ dữ liệu tới các nhân của bộ xử lý sao cho tận dụng được hết khả năng của hệ thống nhúng; Nghiên cứu chung cho tối ưu phần mềm nhúng trên bộ xử lý đa nhân* là rất khó và phức tạp để có thể bao quát được hết các lĩnh vực này.

1.3.2. Mô hình tối ưu phần mềm nhúng trên bộ xử lý đa nhân

Mô hình tối ưu cho phát triển phần mềm nhúng trên bộ xử lý đa nhân như Hình 1.17. Mô hình tối ưu phát triển theo các giai đoạn thiết kế, cài đặt và thực thi. Tuy nhiên, luận án nghiên cứu và xây dựng bài toán tối ưu vào giai đoạn cài đặt với kỹ thuật đảo ngược



Hình 1.17. Mô hình tối ưu phần mềm nhúng trên bộ xử lý đa nhân

1.3.3. Hướng tiếp cận và phương pháp tối ưu phần mềm nhúng trên bộ xử lý đa nhân

Phương pháp đưa ra để tối ưu hiệu năng phần mềm nhúng trên bộ xử lý đa nhân là:

- *Xử lý song song*: Phần mềm nhúng được xây dựng và thực thi trên bộ xử lý đa nhân nên xử lý song song các tác vụ và thực thi song song trên các nhân của bộ xử lý. Các dữ liệu độc lập được phân chia trên các nhân của bộ xử lý để thực thi đồng thời;

- *Phân vùng dữ liệu*: Việc phân vùng dữ liệu giúp cho dữ liệu được phân thành các vùng khác nhau, các vùng dữ liệu này được phân bổ tới các nhân để xử lý;

- *Xử lý bất đồng bộ*: Để làm giảm chi phí đồng bộ trong xử lý song song và đồng thời nhằm tăng hiệu năng thực thi của phần mềm nhúng.

CHƯƠNG 2. TỐI ƯU DỰA TRÊN XỬ LÝ SONG SONG TÁC VỤ

2.1. Tối ưu dựa trên lựa chọn các hàm xử lý chính theo luật Pareto 8/2

2.1.1. Ý tưởng của phương pháp lựa chọn các hàm xử lý chính theo luật Pareto 8/2

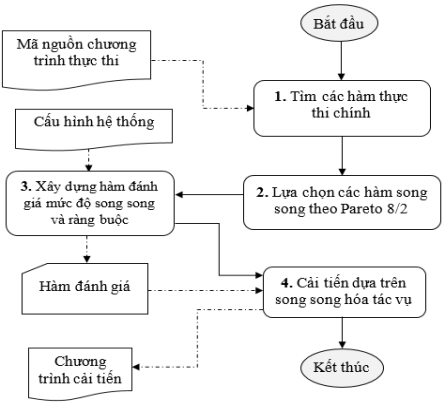
Ý tưởng chính của phương pháp là *dựa trên luật Pareto 8/2 để lựa chọn các hàm xử lý chính; Việc xác định các hàm xử lý chính được thực hiện dựa trên tần suất gọi hàm và số vòng lặp, số dòng lệnh trong hàm; Cần xây dựng hàm điều kiện để xác định số tác vụ chính đưa vào thực hiện việc song song hóa tác vụ.*

2.1.2. Phương pháp đề xuất

2.1.2.1. Mô hình tổng quát

Chúng tôi đề xuất và phát triển phương pháp xử lý song song tác vụ nhằm tối ưu hiệu năng phần mềm nhúng dựa trên ý tưởng lựa chọn hàm xử lý chính theo luật Pareto 8/2. Mô hình tối ưu như Hình 2.1.

Mã nguồn được phân tích để tìm các hàm xử lý chính dựa trên tần suất gọi hàm và số vòng lặp, số dòng lệnh trong hàm; Sau đó, xác định điều kiện để thực hiện việc song song hóa mã nguồn; tối ưu dựa trên song song hóa mã nguồn.



Hình 2.1. Mô hình tối ưu song song tác vụ theo luật Pareto 8/2

2.1.2.2. Điều kiện song song

Giả sử bộ xử lý hệ thống có N nhân đồng nhất, tần số xung nhịp của mỗi nhân là f_x, số xung nhịp trên 1 chu kỳ đồng hồ là f_c, mỗi lệnh được thực hiện trung bình trong C chu kỳ đồng hồ, thời gian truy xuất 1 byte nhớ là T_M. Với mỗi cấu trúc lặp trong hàm, giả sử độ phức tạp (số bước lặp) là M và số câu lệnh trong cấu trúc lặp là N_c. Theo giả thiết này, Gọi S là số lần thực hiện câu lệnh sau khi hoàn thành vòng lặp, thì S được đánh giá như Công thức (2.1).

$$S = M \times N_c \tag{2.1}$$

Thời gian thực thi câu lệnh T_s đánh giá theo Công thức (2.2).

$$T_s = \frac{1}{f_x} \times f_c \times C = \frac{f_c \times C}{f_x} \tag{2.2}$$

Tổng thời gian hoàn thành vòng lặp khi thực hiện trên 1 nhân T_1 được đánh giá theo Công thức (2.3).

$$T_1 = S \times T_s = M \times N_c \times \frac{f_c \times C}{f_x} \tag{2.3}$$

Gọi N_t là số luồng thực thi song song. Thời gian phân luồng và đồng bộ theo thời gian truy xuất bộ nhớ là T và được đánh giá theo Công thức (2.4).

$$T = N_t \times T_M \tag{2.4}$$

Tổng thời gian hoàn thành vòng lặp theo đa luồng T_2 có thể được đánh giá theo Công thức (2.5).

$$T_2 = T + \frac{T_1}{N_t} = N_t \times T_M + \frac{T_1}{N_t} \tag{2.5}$$

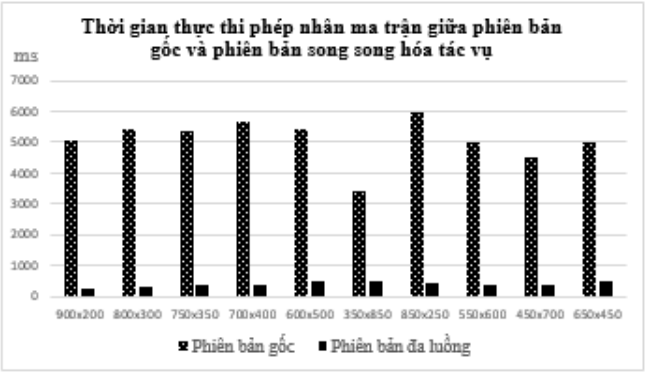
Từ Công thức (2.3) và (2.5) thì điều kiện để thực hiện đa luồng và số luồng song song phải thỏa mãn Hệ bất phương trình 2.6.

$$\begin{cases} T_2 < T_1 \\ N_t \leq N \end{cases} \tag{2.6}$$

2.1.3. Thực nghiệm

Bài toán nhân ma trận được xây dựng để thực nghiệm cho ý tưởng và phát triển phương pháp tối ưu dựa trên kết hợp việc thực thi đồng thời cả mã bytecode trên máy ảo Java và mã máy được biên dịch từ mã nguồn Java trong một ứng dụng Android (phiên bản gốc) cùng với việc song song hóa mã Java cho các bộ xử lý đa nhân (phiên bản đa luồng).

Phương pháp tối ưu đề xuất rút ngắn được khoảng 85% thời gian thực hiện. Điều này được thể hiện rõ trong Hình 2.4.



Hình 2.4. Biểu đồ so sánh thời gian thực thi phép nhân ma trận giữa phiên bản tuần tự và phiên bản song song hóa tác vụ

2.2. Tối ưu cấu hình mã nguồn phần mềm nhúng

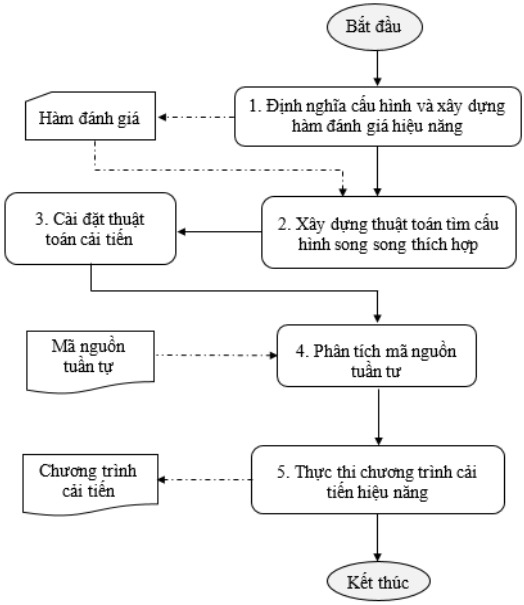
2.2.1. Ý tưởng của phương pháp cấu hình mã nguồn

Ý tưởng chính của phương pháp đề xuất là *tìm cấu hình song song có hiệu năng tốt nhất cho mã nguồn dựa trên đánh giá cấu trúc và bộ tham số tương ứng. Mã nguồn được phân tích để tìm cấu trúc song song ban đầu. Chương trình tối ưu có đầu vào là cấu trúc song song ban đầu và đầu ra là cấu hình tốt nhất.*

2.2.2. Phát triển phương pháp

2.2.2.1. Mô hình tổng quát

Tối ưu cấu hình mã nguồn thực chất là việc tìm được cấu hình song song có hiệu năng tốt nhất do đó cần phải định nghĩa được cấu hình song song cho các bài toán và xây dựng hàm đánh giá hiệu năng trên mỗi cấu hình. Từ đó, luận án đã trình bày mô hình tổng quát của phương pháp như Hình 2.5.



Hình 2.5. Mô hình tổng quát tối ưu cấu hình mã nguồn phần mềm nhúng

2.2.2.2. Xây dựng điều kiện cấu hình

Định nghĩa 2.1: Cấu trúc song song là cấu trúc của một nhóm các chỉ dẫn biên dịch và tham số tương ứng để định nghĩa một đoạn mã được thực hiện song song.

Định nghĩa 2.2: Cấu hình song song của chương trình là một tập hợp các bộ cấu trúc song song và tập tham số tương ứng của cấu trúc. Mỗi chương trình có một tập cấu hình song song như mô tả trong Công thức (2.7).

$$C = \{c \mid c = \{<a, p>\}, a \in A \text{ và } p \in P\} \quad (2.7)$$

Trong đó: C là tập cấu hình và c là một cấu hình song song cụ thể của chương trình; A là tập hợp các cấu trúc song song của các đoạn mã song song trong chương trình, a là cấu trúc của một đoạn mã song song, a bao gồm một tập các chỉ dẫn biên dịch cùng với vị trí trong chương trình; P là tập hợp các bộ tham số, p là bộ tham số tương ứng của mỗi cấu trúc a.

Xây dựng hàm đánh giá hiệu năng

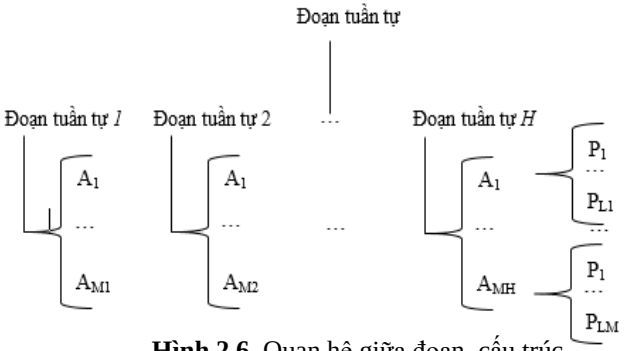
Để đánh giá hiệu năng của chương trình theo mỗi cấu hình, chúng tôi xây dựng hàm đánh giá hiệu năng f như Công thức (2.8).

$$f = \sum_1^H f_i(a_i, p_i) \quad (2.8)$$

Trong đó: H là số đoạn mã có thể song song trong chương trình; f_i là hàm đánh giá hiệu năng trên cấu trúc a_i; cách tính giá trị của f_i có thể khác nhau trên mỗi dạng cấu trúc; a_i là cấu trúc song song thứ i; p_i là bộ tham số tương ứng với cấu trúc a_i.

2.2.2.3. Lựa chọn cấu hình song song tốt nhất

Với chương trình có H đoạn mã có thể song song hóa được, mỗi đoạn mã thứ i có thể có M_i cấu trúc và mỗi cấu trúc j có thể có L_j bộ tham số tương ứng. Mỗi bộ cấu hình của một cấu trúc trong mỗi đoạn có



Hình 2.6. Quan hệ giữa đoạn, cấu trúc và tham số

thể kết hợp với một trong các bộ cấu hình bất kỳ của mỗi cấu trúc trong mỗi đoạn khác như Hình 2.6. Số lượng cấu hình được tính như Công thức (2.9).

$$H_c = \prod_{i=1}^{H} \sum_{j=1}^{M_i} L_j \quad (2.9)$$

Trong đó: H_C là số lượng cấu hình tối đa; M_i là số lượng cấu trúc của đoạn i; L_j là số bộ tham số của cấu trúc thứ j của đoạn i.

Để lựa chọn cấu hình song song tối ưu, trong phạm vi nghiên cứu, chúng tôi sử dụng thuật toán vét cạn. Dựa vào Công thức (2.9), độ phức tạp O của thuật toán được tính theo Công thức (2.10).

$$O = (M_{max} \times L_{max})N \quad (2.10)$$

Trong đó: M_{max} là giá trị lớn nhất của số lượng cấu trúc của các đoạn; L_{max} là giá trị lớn nhất của số lượng bộ tham số của các cấu trúc.

2.2.3. Thực nghiệm

Luận án đề xuất phương pháp lựa chọn cấu hình song song hóa mã nguồn để cải tiến hiệu năng cho phần mềm nhúng trên hệ điều hành Android. Mã nguồn thuần C/C++ được phân tích để tìm ra các cấu trúc song song theo OpenMP và các thông số tương ứng của cấu trúc song song.

Phiên bản gốc ký hiệu là V1 là phiên bản mã nguồn tuần tự C/C++ của chương trình Android; Phiên bản ký hiệu là V2 là tệp apk được biên dịch từ tệp mã nguồn song song được chuyển đổi từ tuần tự sang song song; Phiên bản ký hiệu là V3 là phiên bản được được biên dịch và đóng gói từ tệp mã nguồn đã tìm được cấu hình song song tốt nhất.

Bảng 2.8. Kết quả thực nghiệm

Mã	Thời gian thực thi trung bình (ms)			Tỷ lệ cải tiến hiệu năng (%)		
	V1	V2	V3	V2 so với V1	V3 so với V1	V3 so với V2
P1	42680	45680	44595	-7,03	-4,49	2,38
P2	51280	50860	48355	0,82	5,70	4,93
P3	674937	704937	694937	-4,44	-2,96	1,42
P4	4062780	3668450	3565275	9,71	12,25	2,81
P5	2412840	2336856	2155680	3,15	10,66	7,75
P6	37590	37009	36293	1,55	3,45	1,93
P7	2481304	2258765	2215468	8,97	10,71	1,92
P8	70560	68950	65758	2,28	6,81	4,63
P9	6284260	6216544	6195432	1,08	1,41	0,34
P10	9710776	9155268	9068112	5,72	6,62	0,95
P11	16622	16055	15359	3,41	7,60	4,34
P12	38745	37955	36135	2,04	6,74	4,80
Trung bình				**2,27**	**5,37**	**3,18**

Theo kết quả thực nghiệm trong Bảng 2.8, tỷ lệ cải tiến hiệu năng phụ thuộc vào kích thước và cấu trúc của phiên bản gốc.

2.3 Thảo luận và đánh giá kết quả

Phương pháp lựa chọn các hàm xử lý chính theo luật Pareto 8/2 đạt được kết quả rất tốt với các bài toán xử lý có dùng vòng lặp hay các hàm tác vụ chính có khả năng phân rã được. Phương pháp cải tiến hiệu năng dựa trên lựa chọn hàm và điều kiện ràng buộc đã được được trình bày tại kỷ yếu Fair 2017 [1].

Phương pháp tối ưu cấu hình mã nguồn dựa trên lựa chọn cấu hình song song thích hợp được đề xuất để hỗ trợ và cải tiến cho các phương pháp hiện tại sau khi đã song song hóa mã nguồn. Kết quả nghiên cứu được trình bày tại hội nghị NISC 2017 [2].

CHƯƠNG 3. TỐI ƯU DỰA TRÊN XỬ LÝ SONG SONG DỮ LIỆU

3.1. Phân chia dữ liệu cân bằng và phân bổ động tới các nhân của bộ xử lý

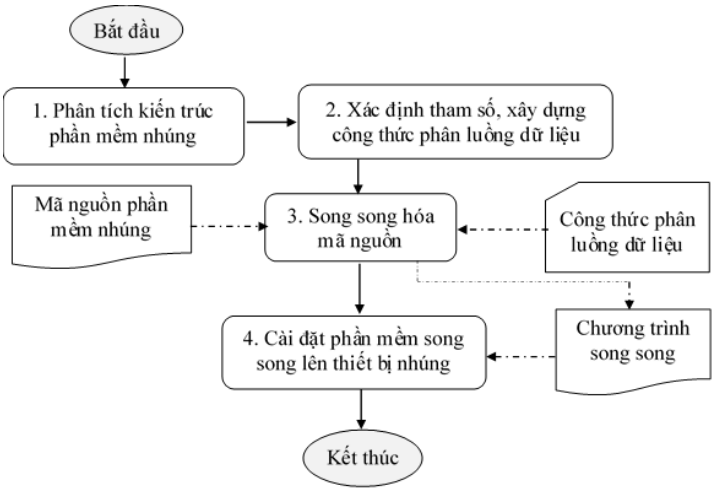
3.1.1. Ý tưởng của phương pháp phân chia dữ liệu cân bằng và phân bổ động tới các nhân của bộ xử lý

Ý tưởng *phân chia dữ liệu cân bằng và phân bổ động nhằm song song hóa dữ liệu với mục đích cải tiến hiệu năng phần mềm nhúng* dựa trên hướng tiếp cận *phân chia các bộ dữ liệu độc lập tới các nhân của bộ xử lý đa nhân và thực hiện xử lý các bộ dữ liệu độc lập đó một cách đồng thời.*

3.1.2. Phương pháp đề xuất

3.1.2.1. Mô hình tổng quát

Để phát triển phương pháp, chúng tôi xây dựng mô hình tổng quát thể hiện quy trình nghiên cứu như trong Hình 3.1.



Hình 3.1. Mô hình tổng quát xử lý song song dữ liệu

3.1.2.2. Điều kiện song song hóa

Tính chất độc lập, phụ thuộc của dữ liệu được mô tả trong các định nghĩa sau.

Định nghĩa 3.1: Bộ dữ liệu độc lập theo một tác vụ là bộ dữ liệu bao gồm các thành phần dữ liệu không ảnh hưởng đến nhau khi thực hiện tác vụ đó.

Định nghĩa 3.2: Bộ dữ liệu phụ thuộc theo một tác vụ là bộ dữ liệu có ít nhất hai thành phần dữ liệu ảnh hưởng đến nhau khi thực hiện tác vụ đó.

Định nghĩa 3.3: Các bộ dữ liệu độc lập là các bộ dữ liệu có mọi thành phần dữ liệu độc lập với nhau và độc lập với các thành phần dữ liệu trong các bộ dữ liệu còn lại.

Giả thiết bộ xử lý có N nhân đồng nhất, gọi d_i là kích thước của một dữ liệu độc lập trong bộ dữ liệu độc lập thứ i, gọi S_i là số dữ liệu trong bộ dữ liệu độc lập thứ i, kích thước bộ dữ liệu độc lập thứ i là D_i được tính theo Công thức (3.1).

$$D_i = S_i \times d_i \tag{3.1}$$

Giả sử K bộ dữ liệu độc lập, thì kích thước dữ liệu D của các bộ dữ liệu độc lập cần được xử lý tính theo Công thức (3.2).

$$D = \sum_1^K D_i = \sum_1^K S_i \times d_i \tag{3.2}$$

Để xác định luồng xử lý dữ liệu cho các bộ dữ liệu độc lập sao cho đạt hiệu năng cao thì cần phân chia luồng theo tiêu chí nhất định, các luồng được chia dựa theo tỷ lệ kích thước dữ liệu cần xử lý của các bộ dữ liệu độc lập với số nhân của bộ xử lý, Gọi N_i là số luồng dữ liệu cần xử lý của bộ thứ i, khi đó N_i được tính theo Công thức (3.3).

$$N_i = \left\lfloor N \times \frac{D_i}{D} = \frac{S_i \times d_i}{\sum_1^K S_i \times d_i} \right\rfloor \tag{3.3}$$

Do S_i là số dữ liệu độc lập nên sau khi xác định số luồng như trong Công thức (3.3), các luồng được phân chia xử lý lượng dữ liệu như nhau. Gọi kích thước dữ liệu được xử lý bởi một luồng là d và d được xác định như trong Công thức (3.4).

$$d = \frac{D}{N} = \frac{\sum_1^K S_i \times d_i}{N} \quad (3.4)$$

3.1.3. Thực nghiệm

Thực nghiệm trên hệ thống IoT sử dụng máy tính nhúng đa nhân Raspberry và giao thức MQTT để kiểm tra và đánh giá hiệu quả của phương pháp phân chia luồng dữ liệu động để xử lý song song hóa dữ liệu.

Không mất tổng quát và đúng với Định nghĩa 3.1, 3.2, 3.3 chúng tôi xét các bộ dữ liệu độc lập là thông điệp đến và thông điệp đi.

Từ Công thức (3.3), phân tích thực nghiệm trên hệ thống IoT sử dụng giao thức MQTT, chúng tôi đưa ra Công thức phân luồng cho thực nghiệm.

$$N_s = \left\lfloor N \times \frac{d_s \times S_s}{d_s \times S_s + d_r \times S_r} \right\rfloor \quad (3.5)$$

$$N_r = \left\lfloor N \times \frac{d_r \times S_r}{d_s \times S_s + d_r \times S_r} \right\rfloor \quad (3.6)$$

Trong đó: N_r là số luồng xử lý dữ liệu đến; N_s là số luồng xử lý dữ liệu đi; S_r là số thông điệp đến; S_s là số thông điệp đi; d_r là kích thước của thông điệp đến; d_s là kích thước của thông điệp đi.

Do S_s, S_r là các bộ dữ liệu độc lập và S_s độc lập với S_r các luồng được phân chia xử lý lượng dữ liệu như nhau. Gọi kích

thước dữ liệu được xử lý bởi một luồng là *d* và *d* được xác định như trong Công thức (3.7).

$$d = \frac{d_s \times S_s + d_r \times S_r}{N} \quad (3.7)$$

Kết quả thực nghiệm được tổng hợp trong Bảng 3.4. Trong bảng này, P1 – P4 là mã các chương trình thực nghiệm, phiên bản gốc tương ứng với các chương trình mã tuần tự và phiên bản cải tiến tương ứng với các chương trình đa luồng. Tỷ lệ cải tiến tương ứng với từng chương trình được tính toán như trong cột cuối. Tỷ lệ cải tiến hiệu năng trung bình là 22,33%.

Bảng 3.4. Tổng hợp kết quả thực nghiệm

Mã chương trình	Thời gian thực thi (ms)		Tỷ lệ cải tiến hiệu năng (%)
	Phiên bản gốc	Phiên bản đa luồng	
P1	6619	7915	-16,37
P2	7169	3495	51,24
P3	132863	68351	48,55
P4	74451	70046	5,91
Average			**22,33**

Kết quả thực nghiệm cho thấy phương pháp cải tiến hiệu năng này được áp dụng hiệu quả cho các bộ xử lý đa nhân, đặc biệt là với các bộ dữ liệu độc lập, kích thước lớn.

3.2 Tối ưu dựa trên phân vùng dữ liệu và xử lý bất đồng bộ

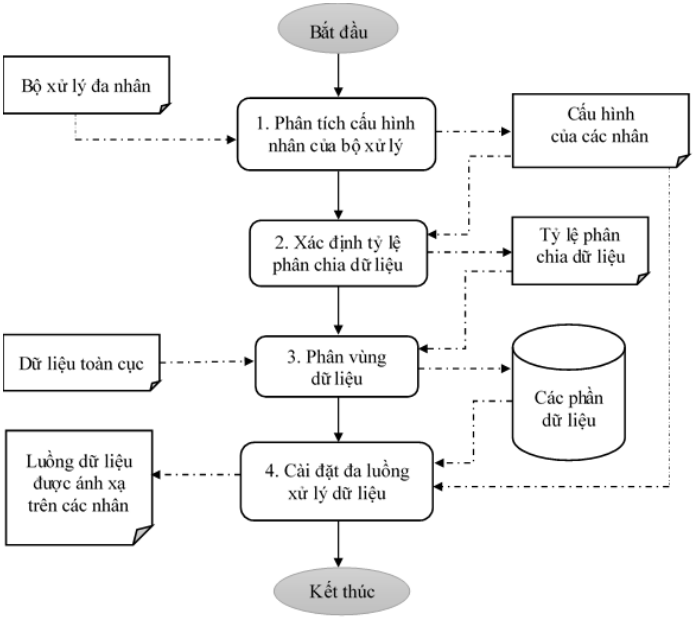
3.2.1. Ý tưởng của phương pháp phân vùng dữ liệu và xử lý bất đồng bộ

Ý tưởng chính của phương pháp đề xuất là *phân chia dữ liệu toàn cục dựa trên cấu hình các nhân của bộ xử lý và xử lý dữ liệu không đồng bộ giữa các luồng để cải tiến hiệu năng*.

3.2.2. Phương pháp đề xuất

3.2.2.1. Mô hình tổng quát

Quy trình phát triển của phương pháp được mô tả như trong mô hình tổng quát như Hình 3.4.



Hình 3.4. Mô hình tổng quát phân chia dữ liệu theo cấu hình nhân CPU và xử lý bất đồng bộ

3.2.2.2. Xác định điều kiện phân chia dữ liệu và phân luồng xử lý

Để tính được tỷ lệ dữ liệu cho từng nhân, luận án đã đưa ra các định nghĩa về tỷ lệ theo tốc độ, khả năng lưu và sự tổng hợp của các nhân.

Định nghĩa 3.4: Tỷ lệ theo tốc độ là tỷ lệ giữa khả năng xử lý của nhân hiện tại trong tập các nhân của vi xử lý. Tỷ lệ này được

sử dụng để phân chia dữ liệu theo tốc độ nhân. Tỷ lệ ký hiệu là r_s, được mô tả trong Công thức (3.8).

$$r_s = \frac{s_i}{\sum_{i=1}^{N} s_i} \quad (3.8)$$

Định nghĩa 3.5: Tỷ lệ theo kích thước bộ nhớ đệm là tỷ lệ lưu trữ dữ liệu trên bộ nhớ đệm của nhân hiện tại trong tập các nhân của bộ xử lý. Tỷ lệ này được sử dụng để phân vùng dữ liệu theo kích thước bộ đệm của nhân. Tỷ lệ ký hiệu là r_c, được mô tả trong Công thức (3.9).

$$r_c = \frac{c_i}{\sum_{i=1}^{N} c_i} \quad (3.9)$$

Định nghĩa 3.6: Tỷ lệ tổng hợp là tỷ lệ của sự kết hợp giữa tỷ lệ tốc độ và tỷ lệ kích thước bộ nhớ đệm của nhân hiện tại trong tập các nhân của bộ xử lý. Tỷ lệ này được sử dụng để phân vùng dữ liệu theo tỷ lệ tổng hợp của nhân. Tỷ lệ ký hiệu là r, được mô tả trong Công thức (3.10).

$$r = \alpha \times r_s + \beta \times r_c = \propto \times \frac{s_i}{\sum_{i=1}^{N} s_i} + \beta \times \frac{c_i}{\sum_{i=1}^{N} c_i} \quad (3.10)$$

Trong đó: s_i là tốc độ của nhân thứ i; N là số nhân của bộ xử lý; c_i là kích thước bộ nhớ đệm của nhân thứ i; N là số nhân của bộ xử lý; α, β là các hệ số tỷ lệ; $\alpha + \beta = 1$.

Từ các định nghĩa trên, phân chia tập dữ liệu D_t ban đầu thành các tập con được thực hiện theo ba cách:

Phân chia dữ liệu tỷ lệ với tốc độ các core/processor

Kích thước dữ liệu Ds_i phân chia theo tỷ lệ tốc độ của thành phần thứ i được xác định dựa trên tỷ lệ tốc độ của nhân thứ i với kích thước dữ liệu cần xử lý và được tính theo Công thức (3.11).

$$Ds_i = r_s \times D_t = \frac{s_i}{\sum_{i=1}^{N} s_i} \times D_t \quad (3.11)$$

Phân chia dữ liệu tỷ lệ với kích thước bộ đệm

Kích thước dữ liệu Dc_i phân chia theo tỷ lệ kích thước bộ nhớ đệm của thành phần thứ *i* được xác định dựa trên tỷ lệ kích thước bộ nhớ đệm của nhân thứ *i* với kích thước dữ liệu cần xử lý và tính theo Công thức (3.16).

$$Dc_i = r_s \times D_t = \frac{s_i}{\sum_{i=1}^{N} s_i} \times D_t \quad (3.16)$$

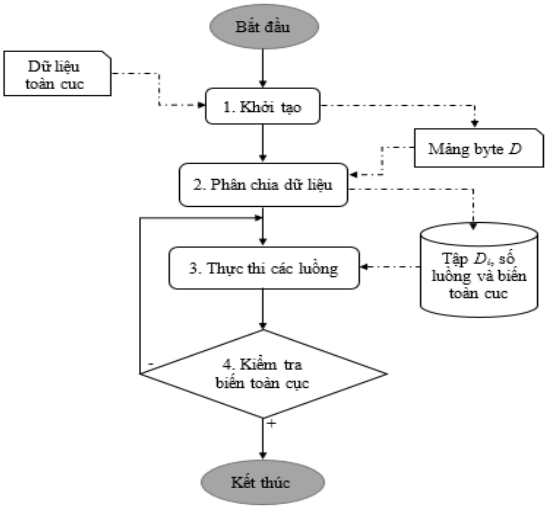
Phân chia sử dụng kết hợp 2 tham số tốc độ và kích thước bộ đệm

Kích thước dữ liệu Dr_i phân chia theo tỷ lệ tổng hợp của thành phần thứ *i* được xác định dựa trên tỷ lệ tổng hợp của nhân thứ *i* với kích thước dữ liệu cần xử lý, được tính theo Công thức (3.17).

$$Dr_i = r \times D_t = (\alpha \times r_s + \beta \times r_c) \times D_t \quad (3.17)$$

3.2.2.3. Xây dựng mô hình đa luồng xử lý bất đồng bộ để tối ưu

Để giải quyết bài toán bất đồng bộ luồng khi độc lập dữ liệu, phương pháp được đề xuất các tổ chức dữ liệu đầu vào, đầu ra và biến đếm trong



Hình 3.5. Mô hình xử lý bất đồng bộ

vùng nhớ toàn cục như Hình 3.5.

Vì là vùng nhớ toàn cục nên tất cả các luồng đều truy xuất được. Đồng thời, biến đếm toàn cục sẽ cho biết tất cả các luồng đã hoàn thành hay chưa.

3.2.3. Thực nghiệm

Để đánh giá phương pháp đề xuất, chúng tôi tiến hành thực nghiệm trên mô hình hệ thống và thực thi các thuật toán mã hoá AES, DES, 3DES, v.v. cài đặt theo mô hình đơn luồng và mô hình đa luồng với cách phân chia dữ liệu theo các Công thức (3.11), (3.16), (3.17).

Trên Raspberry, thuật toán AES, DES, Triple DES (3DES) được cài đặt để mã hóa video theo mô hình đơn luồng và theo mô hình đa luồng dựa trên phương pháp đề xuất.

Sau khi tiến hành thực nghiệm, chúng tôi nhận được kết quả như Bảng 3.9 dưới đây:

Bảng 3.9. So sánh tỷ lệ cải tiến giữa ba thuật toán mã hóa AES, DES và 3DES

Kích thước dữ liệu (MB)	Tỷ lệ cải tiến (%)		
	AES	DES	3DES
1	44,26	51,19	64,18
2	48,19	56,92	66,27
4	51,07	59,6	67,63
8	52,47	61,12	67,82
16	54,4	61,32	63,67
32	54,42	60,93	68,05
64	57,09	62,03	68,23
Average	**51,78**	**57,59**	**66,55**

Kết quả thực nghiệm được tổng hợp cho thấy: kết quả thực nghiệm với thuật toán AES là 51,78%, DES là 57,59% và Triple DES là 66,55%.

3.3 Thảo luận và đánh giá kết quả

Phương pháp phân chia dữ liệu cân bằng và phân bổ động thực hiện xử lý trên sự độc lập dữ liệu, công thức phân luồng đã được xây dựng để hỗ trợ phân luồng động nhằm chuyển đổi mã nguồn tuần tự sang đa luồng theo sự phân chia dữ liệu cân bằng. Kết quả nghiên cứu đã cho thấy hiệu năng phần mềm nhúng đã được cải tiến tốt hơn so với phần mềm nhúng ban đầu và được trình bày tại hội thảo ATC 2018 [3].

Phương pháp cải thiện hiệu suất phần mềm nhúng trên bộ xử lý đa nhân dựa trên phân vùng dữ liệu và xử lý bất đồng bộ hoạt động tốt, đặc biệt khi kích thước dữ liệu lớn, tỷ lệ cải thiện hiệu suất cao so với các nghiên cứu trước đây đã được chỉ ra. Phương pháp nghiên cứu đã được chấp nhận đăng tại tạp chí “Informatics and Automation journal” theo công trình [4].

KẾT LUẬN

Những kết quả đạt được

- *Vấn đề tối ưu dựa trên xử lý song song tác vụ*, chúng tôi đã nghiên cứu theo hướng *lựa chọn hàm chính theo luật Pareto 8/2* và *tối ưu cấu hình mã nguồn phần mềm nhúng* nhằm tăng hiệu năng của phần mềm nhúng dựa trên việc tận dụng đa nhân của bộ xử lý để thực thi nhiều tác vụ cùng lúc. Về mặt lý thuyết, luận án đã trình bày ý tưởng, đề xuất phát triển phương pháp theo mô hình tổng quát và đưa ra điều kiện song song cũng như các đánh giá cho các điều kiện song song này. Về thực nghiệm, từ mô hình tổng quát, chúng tôi xây dựng mô hình thực nghiệm và triển khai thực nghiệm để chứng minh các lý thuyết đưa ra là đúng.

+ Phương pháp đề xuất *lựa chọn hàm chính theo luật Pareto 8/2 và xác định số hàm tham gia chuyển hóa dựa trên việc phân chia tác vụ chính thực hiện song song tới từng nhân thỏa mãn điều kiện song song* , với ý tưởng chính là *phân chia chức năng tổng thể thành các tác vụ song song, phân phối tác vụ vào các nhân rỗi, lập lịch tác vụ và quản lý trao đổi thông tin giữa các tác vụ để xác định các tác vụ chính và dựa trên hàm điều kiện để xác định số tác vụ thực thi song song hóa*. Phương pháp được công bố trong công trình [1] tại danh mục công trình nghiên cứu.

+ Phương pháp đề xuất *tối ưu cấu hình mã nguồn phần mềm nhúng*, chúng tôi đưa ra ý tưởng *tìm cấu hình song song có hiệu năng tốt nhất cho mã nguồn dựa trên đánh giá cấu trúc và bộ tham số tương ứng* và xây dựng mô hình tổng quát để triển khai ý tưởng. Phương pháp tối ưu dựa trên cấu hình mã nguồn được công bố tại công trình [2] trong danh mục công trình nghiên cứu.

- *Vấn đề song song hóa dữ liệu* là một giải pháp để cải tiến hiệu năng của các phần mềm có các bộ dữ liệu độc lập hay các bộ dữ liệu có thể phân chia để thực hiện xử lý dữ liệu một cách độc lập, và từ các dữ liệu độc lập này sẽ được đưa vào xử lý trên các nhân của bộ xử lý đa nhân của hệ thống nhúng. Về mặt lý thuyết, luận án đã trình bày ý tưởng, đề xuất phát triển phương pháp theo mô hình tổng quát và đưa ra điều kiện song song cũng như các đánh giá cho các điều kiện song song này. Về thực nghiệm, từ mô hình tổng quát, chúng tôi xây dựng mô hình thực nghiệm và triển khai thực nghiệm để chứng minh các lý thuyết đưa ra là đúng.

+ Phương pháp đề xuất *xử lý phân chia dữ liệu cân bằng và phân bổ động* với ý tưởng chính là *phân chia dữ liệu độc lập và phân bổ tới các nhân của bộ xử lý để thực hiện đồng thời xử lý dữ liệu độc lập với mục đích cải tiến hiệu năng phần mềm nhúng*, kết quả nghiên cứu được công bố trong công trình [3] tại danh mục công trình nghiên cứu.

+ Phương pháp đề xuất *tối ưu dựa trên phân vùng dữ liệu và xử lý bất đồng bộ*, với ý tưởng chính là *xây dựng mô hình phân chia dữ liệu toàn cục dựa trên cấu hình các nhân của bộ xử lý và xử lý dữ liệu không đồng bộ giữa các luồng để cải tiến hiệu năng*, kết quả của phương pháp đang gửi trên tạp chí "Informatics and Automation journal" theo công trình [4] trong danh mục công trình nghiên cứu.

Những hạn chế và hướng nghiên cứu tiếp theo

Luận án thực hiện mục tiêu tối ưu phần mềm nhúng trên bộ xử lý đa nhân, tuy nhiên vẫn đang tập trung nghiên cứu và thực

hiện theo một mục tiêu là tối ưu hiệu năng của phần mềm nhúng. Nghiên cứu vẫn còn một số hạn chế như chưa xét đến sự song song hóa về mặt tác vụ hay song song dựa trên kết hợp phân chia tác vụ với phân chia dữ liệu, chưa đánh giá về thời gian trễ phân luồng, chỉ tối ưu cục bộ mô-đun phần mềm trên thiết bị nhúng mà chưa tối ưu tổng thể toàn hệ thống

Các nghiên cứu về xử lý song song đã được nghiên cứu nhiều và nghiên cứu từ trước nên các kết quả chỉ mang tính khẳng định hiệu năng của phần mềm nhúng khi được song song hóa trên các bộ xử lý đa nhân sẽ tốt hơn so với phần mềm nhúng chỉ thực hiện tuần tự.

Nghiên cứu về lựa chọn cấu hình tốt nhất thì hàm đánh giá f_i vẫn chưa được xác định tổng quát mà chỉ được tìm theo từng bài toán cụ thể. Còn trong nghiên cứu về phân vùng dữ liệu và xử lý bất đồng bộ thì chỉ tính đến cấu hình của từng nhân tại thời điểm đầu vào chứ không xác định động trong quá trình thực hiện phần mềm nhúng.

Hướng nghiên cứu tiếp theo, chúng tôi sẽ nghiên cứu, tìm được hàm đánh giá f_i tổng quát để đưa ra lựa chọn cấu hình song song tốt nhất và tiếp tục cải tiến phương pháp ở một số khía cạnh: mở rộng vấn đề để xử lý dữ liệu đồng bộ; ánh xạ các luồng thực thi tới các nhân tương ứng; giám sát và thay đổi việc phân chia dữ liệu theo hiệu suất tại mỗi thời điểm khi mỗi nhân CPU xử lý xong dữ liệu.