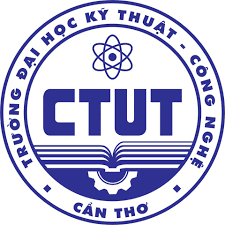
TRƯỜNG ĐẠI HỌC KỸ THUẬT – CÔNG NGHỆ CẦN THƠ

**KHOA CÔNG NGHỆ THÔNG TIN**



**TÍNH TOÁN SONG SONG VÀ PHÂN TÁN**

**Ngành: Công nghệ thông tin**

**BÁO CÁO**

**TÍNH TOÁN PHÂN TÁN**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn:  **ThS. LÊ ANH NHÃ UYÊN** | Sinh viên thực hiện:  **1. HUỲNH TẤN KHƯƠNG - 2101499**  **2. QUÁCH HIẾU KHANG - 2101344**  **3. NGUYỄN GIA HƯNG - 2100065**  **4. LÊ ĐĂNG QUANG - 2100770**  **5. TRẦN QUỐC NHÂN - 2100328**  **6. PHAN THANH NHÃ - 2100688** |

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

…………………………………………………………………………………………..…………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..………………………………………………………………………………………..…………………………………………………………………………

**LỜI CAM ĐOAN**

Chúng em xin cam đoan đề tài “Song song dựa trên tiến trình” được thực hiện bởi 6 thành viên bao gồm Huỳnh Tấn Khương, Quách Hiếu Khang, Nguyễn Gia Hưng, Lê Đăng Quang, Trần Quốc Nhân, Phan Thanh Nhã, là công trình nghiên cứu của riêng nhóm dưới sự hướng dẫn của ThS. Lê Anh Nhã Uyên. Mọi số liệu và kết quả công bố được trình bày trong đề tài đều được trình bày một cách trung thực và mọi tham khảo được sử dụng trong đề tài đều được trích dẫn rõ ràng. Những nội dung trình bày trong quyển báo cáo này không phải là bản sao chép từ bất kì quyển báo cáo, tiểu luận nào có trước. Nếu phát hiện bất kỳ gian dối nào, nhóm em xin chịu hoàn toàn trách nhiệm.

*Cần Thơ, ngày ... tháng ... năm …*

**Sinh viên thực hiện**

**LỜI CẢM ƠN**

Đầu tiên, nhóm xin chân thành cảm ơn các thầy các cô thuộc Trường Đại học Kỹ thuật – Công nghệ Cần Thơ đã trang bị kiến thức cho chúng em trong suốt quá trình học tập tại trường, nhờ sự chỉ bảo tận tình của các thầy các cô mà chúng em mới có được những kiến thức chuyên ngành công nghệ thông tin để có thể thực hiện tiếp chặng đường học tập, vận dụng và sáng tạo ra những sản phẩm hữu ích góp phần phục vụ các lĩnh vực khác nhau.

Thứ hai, nhóm muốn gửi lời cảm ơn chân thành đến cô ThS. Lê Anh Nhã Uyên, người đã tận tình hướng dẫn nhóm hoàn thành bài báo cáo này. Cô đã luôn nhiệt tình giúp đỡ và hướng dẫn về kiến thức cũng như hỗ trợ tinh thần từ khi bắt đầu cho đến khi hoàn thành. Nhờ sự hỗ trợ của Cô, nhóm đã hoàn thành đúng thời hạn và cũng đã tích lũy được nhiều kiến thức quý báu.

Mặc dù, còn nhiều khó khăn hạn chế trong quá trình thực hiện cũng như chưa thành thạo về các kĩ năng, chưa nắm vững được kiến thức chuyên sâu về môn học. Tuy không thể phát huy hết những ý tưởng của Cô, nhưng nhóm em đã tích lũy thêm nhiều kiến thức để có một cái nhìn sâu sắc và hoàn thiện hơn trong học tập. Nhóm em rất mong nhận được sự thông cảm và đóng góp chỉ bảo từ Cô để nhóm rút kinh nghiệm và hoàn thành tốt hơn.

Nhóm xin chúc quý Cô sẽ luôn khỏe mạnh, hoàn thành tốt công việc của mình và luôn hạnh phúc.

*Nhóm xin chân thành cảm ơn!*

*Cần Thơ, ngày tháng năm*

**MỤC LỤC**

**[CHƯƠNG I: TỔNG QUAN 8](#_heading=h.gjdgxs)**

[1.1. Lý do chọn đề tài 8](#_heading=h.30j0zll)

[1.2. Mục tiêu thực hiện 9](#_heading=h.1fob9te)

*[1.2.1 Nghiên cứu lý thuyết 9](#_heading=h.3znysh7)*

*[1.2.2 Khảo sát các mô hình hệ thống phân tán 9](#_heading=h.2et92p0)*

*[1.2.3 Đánh giá và so sánh 9](#_heading=h.tyjcwt)*

*[1.2.4 Nghiên cứu ứng dụng thực tiễn 9](#_heading=h.3dy6vkm)*

*[1.2.5 Đề xuất giải pháp 9](#_heading=h.1t3h5sf)*

[1.3. Lợi ích và thách thức của tính toán phân tán 10](#_heading=h.4d34og8)

[1.4. Bố cục đề tài 10](#_heading=h.2s8eyo1)

**[CHƯƠNG II: CƠ SỞ LÝ THUYẾT 12](#_heading=h.17dp8vu)**

[2.2 Các mô hình của hệ thống phân tán 14](#_heading=h.3rdcrjn)

*[2.2.1 Mô hình máy tính tầm trung (Minicomputer Model) 14](#_heading=h.26in1rg)*

*[2.2.2 Mô hình trạm làm việc (Workstation Model) 15](#_heading=h.35nkun2)*

*[2.2.3 Mô hình trạm làm việc - máy phục vụ (Workstation - Server Model) 16](#_heading=h.44sinio)*

*[2.2.4 Mô hình vùng các bộ xử lý (Processor-Pool Model) 17](#_heading=h.z337ya)*

*[2.2.5 Mô hình kết hợp (Hybrid Model) 18](#_heading=h.1y810tw)*

**[CHƯƠNG III: ỨNG DỤNG SỬ DỤNG PYRO4 29](#_heading=h.1ci93xb)**

[3.1 Tổng quan về Pyro4: 30](#_heading=h.3whwml4)

[3.2 Cách hoạt động: 30](#_heading=h.2bn6wsx)

*[3.2.1 File Serve.py 30](#_heading=h.qsh70q)*

*[3.2.2 File Client.py 31](#_heading=h.3as4poj)*

[3.3. Name Server 32](#_heading=h.1pxezwc)

**[CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 34](#_heading=h.49x2ik5)**

[4.1. Kết luận 34](#_heading=h.2p2csry)

[4.2. Hướng phát triển 35](#_heading=h.147n2zr)

*[4.2.1. Tích hợp với các công nghệ mới 35](#_heading=h.3o7alnk)*

*[4.2.2. Tăng cường bảo mật và quản lý tài nguyên 35](#_heading=h.23ckvvd)*

*[4.2.3. Phát triển các công cụ hỗ trợ mới 35](#_heading=h.ihv636)*

*[4.2.4. Ứng dụng trong các lĩnh vực mới 35](#_heading=h.32hioqz)*

*[4.2.5. Hướng tới các hệ thống phi tập trung và blockchain 36](#_heading=h.1hmsyys)*

[4.3. Kết luận và Tầm nhìn tương lai 36](#_heading=h.41mghml)

**DANH MỤC HÌNH ẢNH**

*[Hình 2.1 Mô hình máy tính tầm trung 15](#_heading=h.lnxbz9)*

*[Hình 2.2 Mô hình trạm làm việc 16](#_heading=h.1ksv4uv)*

*[Hình 2.3 Mô hình trạm làm việc - máy phục vụ 17](#_heading=h.2jxsxqh)*

*[Hình 2.4 Mô hình vùng các bộ xử lý 18](#_heading=h.3j2qqm3)*

*[Hình 2.5 Mô hình kết hợp 19](#_heading=h.4i7ojhp)*

*[Hình 2.6 Mô hình Celery 22](#_heading=h.2xcytpi)*

*[Hình 2.7 Mô hình SCOOP 24](#_heading=h.qsh70q)*

*[Hình 2.8 Mô hình pyro4 26](#_heading=h.3o7alnk)*

*[Hình 3.1 Code file Serve.py 28](#_heading=h.41mghml)*

*[Hình 3.2 Code file Client.py 28](#_heading=h.2grqrue)*

*[Hình 3.3 Khởi tạo Server 31](#_heading=h.19c6y18)*

*[Hình 3.4 Kết quả trả về của Server 31](#_heading=h.3tbugp1)*

*[Hình 3.5 Kết quả trả về của Client 32](#_heading=h.28h4qwu)*

**DANH MỤC BẢNG BIỂU**

*[Bảng 2.1 So sánh song song và phân tán 19](#_heading=h.vnxsfkckynni)*

# CHƯƠNG I: TỔNG QUAN

## 1.1. Lý do chọn đề tài

Trong bối cảnh hiện nay, công nghệ thông tin đã trở thành xương sống của mọi lĩnh vực trong đời sống như **y tế**, **kinh tế**, **giáo dục**, và **khoa học kỹ thuật**. Mặc dù khả năng xử lý của phần cứng và phần mềm không ngừng tăng, nhưng sự bùng nổ của **dữ liệu lớn** (Big Data) và **trí tuệ nhân tạo** (AI) đặt ra những thách thức mới về hiệu năng và thời gian xử lý.

Một số bài toán khoa học như **dự báo thời tiết**, **dự báo biến đổi khí hậu**, **phân tích dữ liệu gen**, hoặc **mô phỏng vật lý phức tạp** yêu cầu năng lực tính toán vượt xa khả năng của một máy tính đơn lẻ. Ngay cả các hệ thống máy tính mạnh mẽ cũng gặp khó khăn khi cần giải quyết các vấn đề trong thời gian thực và yêu cầu bộ nhớ lớn hơn khả năng sẵn có.

Vì vậy, **tính toán phân tán** đã nổi lên như một phương pháp hiệu quả, chia nhỏ bài toán lớn thành nhiều tác vụ nhỏ và phân phối cho nhiều máy tính hoặc nút tính toán khác nhau. Quá trình này giúp:

**- Tăng tốc độ xử lý** bằng cách khai thác song song các tài nguyên.

**- Nâng cao khả năng chịu lỗi**, vì nếu một máy gặp sự cố, các máy khác vẫn hoạt động.

**- Tận dụng tài nguyên phi cục bộ**, giảm chi phí đầu tư vào phần cứng cao cấp.

Ngoài ra, xu hướng sử dụng **cloud computing** (điện toán đám mây) và **edge computing** (điện toán biên) cũng cho thấy nhu cầu ngày càng lớn về tính toán phân tán. Các doanh nghiệp và tổ chức đang chuyển đổi từ mô hình tập trung sang các hệ thống phân tán để đáp ứng nhu cầu phát triển ứng dụng thông minh và quản lý dữ liệu hiệu quả hơn.

## 1.2. Mục tiêu thực hiện

Báo cáo này hướng đến việc cung cấp cái nhìn sâu sắc và toàn diện về **tính toán phân tán**, từ lý thuyết cơ bản đến các ứng dụng thực tế. Các mục tiêu cụ thể bao gồm:

### 1.2.1 Nghiên cứu lý thuyết

- Phân biệt tính toán phân tán với tính toán song song.

- Phân tích các yếu tố cốt lõi trong hệ thống phân tán như tính trong suốt, khả năng mở rộng, và bảo mật.

### 1.2.2 Khảo sát các mô hình hệ thống phân tán

- Mô hình **máy tính tầm trung**.

- Mô hình **trạm làm việc** và **trạm làm việc - máy phục vụ**.

- Mô hình **vùng các bộ xử lý** và **mô hình kết hợp**.

### 1.2.3 Đánh giá và so sánh

- Đánh giá hiệu suất của các hệ thống phân tán so với hệ thống truyền thống.

- So sánh với các phương pháp khác như **thread-based parallelism** để làm rõ ưu và nhược điểm.

### 1.2.4 Nghiên cứu ứng dụng thực tiễn

- Ứng dụng Celery để quản lý tác vụ phân tán và RabbitMQ như một message broker.

- Phân tích hiệu quả của **SCOOP** trong các bài toán khoa học và **Pyro4** trong phát triển hệ thống client-server.

### 1.2.5 Đề xuất giải pháp

- Phân tích các thách thức khi triển khai hệ thống phân tán như bảo mật và quản lý tài nguyên.

- Đề xuất các hướng phát triển để cải thiện hiệu suất và khả năng mở rộng của hệ thống trong tương lai.

## 1.3. Lợi ích và thách thức của tính toán phân tán

**Lợi ích:**

**- Tăng tốc độ xử lý:** Chia bài toán lớn thành nhiều phần nhỏ giúp xử lý đồng thời.

**- Tính linh hoạt:** Có thể sử dụng nhiều thiết bị khác nhau như máy tính, điện thoại thông minh, hoặc cảm biến.

**- Giảm chi phí:** Tận dụng tài nguyên sẵn có thay vì đầu tư vào máy chủ lớn.

**- Khả năng chịu lỗi:** Hệ thống tiếp tục hoạt động ngay cả khi một phần gặp sự cố.

**Thách thức:**

**- Bảo mật:** Dữ liệu phải được bảo vệ trong môi trường phân tán.

**- Đồng bộ hóa:** Cần đồng bộ giữa các máy tính để đảm bảo tính nhất quán dữ liệu.

**- Quản lý tài nguyên:** Phân phối tài nguyên hợp lý để tránh lãng phí hoặc quá tải.

**- Độ phức tạp trong triển khai:** Yêu cầu kiến thức chuyên sâu về lập trình phân tán và các giao thức truyền thông.

## 1.4. Bố cục đề tài

Bài báo cáo này gồm 4 chương chính, được trình bày như sau:

**Chương 1:** Tổng quan.

- Giới thiệu lý do chọn đề tài, mục tiêu và lợi ích của tính toán phân tán.

**Chương 2:** Cơ sở lý thuyết.

- Phân tích chi tiết các khái niệm, mô hình tính toán phân tán và công cụ hỗ trợ.

**Chương 3:** Ứng dụng.

- Minh họa ứng dụng tính toán phân tán với Celery, SCOOP và Pyro4.

**Chương 4:** Kết luận và hướng phát triển.

- Đánh giá những điểm mạnh và yếu, cùng đề xuất cải tiến hệ thống trong tương lai.

Phần cuối cùng là tài liệu tham khảo, tập hợp các nguồn tài liệu và công trình nghiên cứu hỗ trợ cho đề tài.

# CHƯƠNG II: CƠ SỞ LÝ THUYẾT

**2.1 [Tìm hiểu về tính toán phân tán (Distributed Computing)](#_heading=h.37m2jsg)**

Khái niệm, lý thuyết tính toán phân tán:

Tính toán phân tán(Distributed Computing) là một lĩnh vực trong khoa học máy tính liên quan đến việc phân chia và thực hiện các tác vụ tính toán trên nhiều máy tính hoặc nút (node) khác nhau, kết nối với nhau qua mạng. Các máy này có thể hoạt động song song để hoàn thành một nhiệm vụ lớn mà một máy tính đơn lẻ khó có thể thực hiện hiệu quả hoặc nhanh chóng.

Lý thuyết của tính toán phân tán (Distributed Computing) tập trung vào các nguyên tắc và cơ chế để tổ chức, điều phối, và quản lý các hệ thống phân tán nhằm tối ưu hóa hiệu suất, đảm bảo tính tin cậy và nhất quán trong khi xử lý các nhiệm vụ tính toán trên nhiều máy tính độc lập.

**Ưu điểm:**

* Mô hình phân tán: Nhiều máy tính độc lập, không chia sẻ bộ nhớ, giao tiếp qua mạng.
* Truyền thông: Sử dụng RPC, giao tiếp thông điệp, REST API; gặp vấn đề về độ trễ và băng thông.
* Đồng bộ hóa và nhất quán: Đảm bảo dữ liệu nhất quán giữa các nút, dùng giao thức Paxos, Raft.
* Khả năng chịu lỗi: Hệ thống vẫn hoạt động khi một số nút hỏng, dùng phân phối lại công việc.
* Tính mở rộng: Thêm nhiều nút mà không giảm hiệu suất, dùng kiến trúc chia để trị.
* Tính minh bạch: Người dùng không nhận biết các máy tính phân tán.
* CAP Theorem: Chọn hai trong ba thuộc tính: nhất quán, khả dụng, khả năng chịu lỗi phân vùng.
* Thuật toán đồng thuận: Đảm bảo các nút đồng ý về trạng thái hệ thống, dùng Paxos, Raft.

*Bảng 2.1 So sánh song song và phân tán*

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Tính toán song song** | **Tính toán phân tán** |
| **Môi trường triển khai** | Trong cùng một máy tính, bộ nhớ dùng chung | Nhiều máy tính độc lập, kết nối qua mạng |
| **Quản lý tài nguyên** | Bộ nhớ và CPU chung | Tài nguyên riêng biệt trên mỗi máy tính |
| **Đồng bộ hóa** | Dễ hơn, qua bộ nhớ dùng chung | Tài nguyên riêng biệt trên mỗi máy tính |
| **Tính chịu lỗi** | Ít chịu lỗi, dễ ảnh hưởng nếu một phần bị lỗi | Chịu lỗi tốt hơn, có thể tiếp tục hoạt động nếu một nút hỏng |
| **Khả năng mở rộng** | Giới hạn theo số lõi CPU/GPU của hệ thống | Dễ mở rộng, có thể thêm nhiều máy tính |
| **Tốc độ giao tiếp** | Nhanh, trong cùng hệ thống | Chậm hơn, phải qua mạng |
| **Ứng dụng** | Tính toán khoa học, xử lý đồ họa, mô phỏng vật lý | Điện toán đám mây, xử lý dữ liệu lớn, blockchain |

Khái niệm, lý thuyết hệ thống phân tán:  
Hệ thống phân tán (Distributed System) là một mô hình trong đó các thành phần phần mềm hoặc phần cứng nằm trên nhiều máy tính, kết nối với nhau qua mạng, cùng phối hợp để đạt được một mục tiêu chung. Từ góc nhìn của người dùng hoặc chương trình, hệ thống phân tán hoạt động như một hệ thống đơn nhất, dù thực tế nó được cấu thành từ nhiều hệ thống độc lập.  
 Lý thuyết của hệ thống phân tán tập trung vào các khía cạnh liên quan đến việc thiết kế, triển khai và quản lý các hệ thống máy tính hoạt động đồng thời trên nhiều máy tính hoặc nút độc lập, nhằm đạt được mục tiêu chung.

**Ưu điểm:**

* Mô hình hệ thống: Nhiều máy tính độc lập, không chia sẻ bộ nhớ, kết nối qua mạng.
* Truyền thông: Giao tiếp qua mạng với giao thức như RPC, Message Passing.
* Đồng bộ hóa: Đồng bộ dữ liệu và trạng thái giữa các nút, dùng các thuật toán như Lamport, Paxos.
* Nhất quán dữ liệu: Đảm bảo dữ liệu nhất quán toàn hệ thống, liên quan đến CAP Theorem.
* Tính chịu lỗi: Hệ thống vẫn hoạt động khi một số nút bị lỗi, dùng cơ chế sao lưu và phục hồi.
* Tính mở rộng: Hệ thống dễ mở rộng bằng cách thêm nút mới mà không ảnh hưởng đến toàn hệ thống.
* Tính minh bạch: Người dùng không cần biết hệ thống phân tán trên nhiều máy tính khác nhau.
* An ninh và bảo mật: Quản lý quyền truy cập, mã hóa dữ liệu để bảo vệ trong môi trường phân tán.

## 2.2 Các mô hình của hệ thống phân tán

### ***2.2.1 Mô hình máy tính tầm trung (Minicomputer Model)***

Mô hình này sử dụng một số máy tính tầm trung, được kết nối qua hệ thống mạng. Mỗi máy có thể phục vụ nhiều người dùng đồng thời, cho phép người dùng truy cập từ xa để sử dụng tài nguyên của máy khác. Ví dụ nổi bật của mô hình này là ARPANet, mạng lưới truyền thông được phát triển cho quân đội Mỹ, tiền thân của Internet ngày nay.

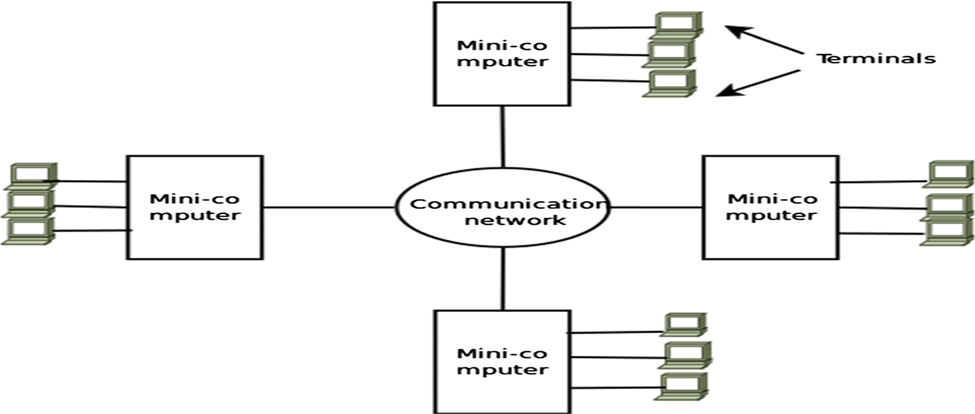
**Ưu điểm**:

- Khả năng truy cập tài nguyên từ xa.

- Hỗ trợ nhiều người dùng truy cập đồng thời.

**Hạn chế**:

- Sự phức tạp trong việc quản lý và bảo trì hệ thống lớn.



*Hình 2.1 Mô hình máy tính tầm trung*

**Hạn chế**:

- Sự phức tạp trong việc quản lý và bảo trì hệ thống lớn.

### ***2.2.2 Mô hình trạm làm việc (Workstation Model)***

Trong mô hình này, mỗi trạm làm việc là một máy tính riêng biệt, trang bị đĩa cứng và phục vụ một người dùng duy nhất. Các trạm làm việc được liên kết qua mạng tốc độ cao, giúp các trạm quá tải có thể mượn tài nguyên từ các trạm rảnh.

**Ưu điểm**:

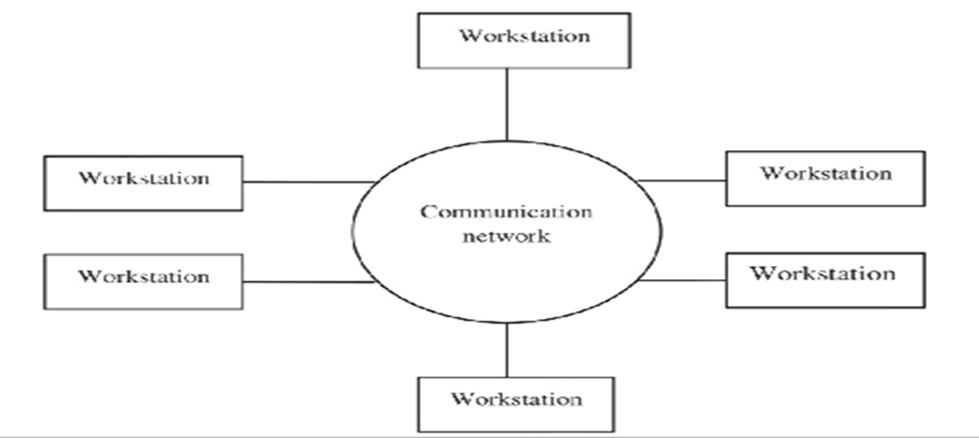
- Mô hình phân bổ rõ ràng, mỗi người dùng có máy tính riêng.

- Linh hoạt trong việc tận dụng tài nguyên từ các trạm khác.

**Hạn chế**:

- Lãng phí tài nguyên khi một số trạm không hoạt động.

- Chi phí cao do cần trang bị đĩa cứng cho mỗi trạm.



*Hình 2.2 Mô hình trạm làm việc*

### ***2.2.3 Mô hình trạm làm việc - máy phục vụ (Workstation - Server Model)***

Mô hình này kết hợp trạm làm việc và máy phục vụ. Các trạm làm việc có thể không có đĩa cứng, nhưng chúng có thể truy cập tài nguyên từ máy phục vụ trung tâm qua mạng. Điều này giúp giảm chi phí và dễ dàng hơn trong quản lý hệ thống.

**Ưu điểm**:

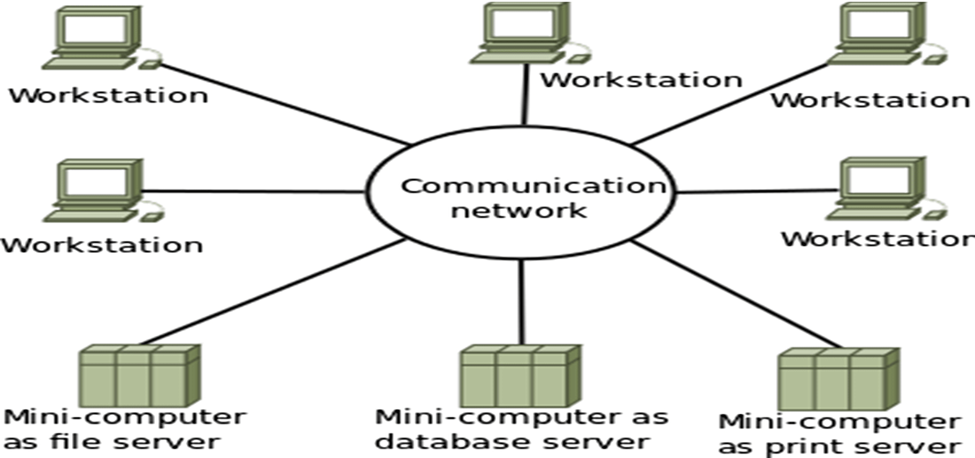
- Giảm chi phí phần cứng do không cần trang bị đĩa cứng cho tất cả các trạm.

- Dễ bảo trì và quản lý dữ liệu tập trung.

**Hạn chế**:

- Không tận dụng tối đa năng lực xử lý của các trạm làm việc rảnh rỗi.

- Có thể gặp tình trạng quá tải ở máy phục vụ trung tâm.



*Hình 2.3 Mô hình trạm làm việc - máy phục vụ*

### ***2.2.4 Mô hình vùng các bộ xử lý (Processor-Pool Model)***

Mô hình này được xây dựng dựa trên việc gộp chung tất cả các bộ xử lý thành một vùng và phân phối chúng theo yêu cầu. Người dùng có thể yêu cầu bộ xử lý để thực hiện các tác vụ lớn khi cần thiết. Ví dụ về hệ thống này bao gồm Amoeba và Plan 9.

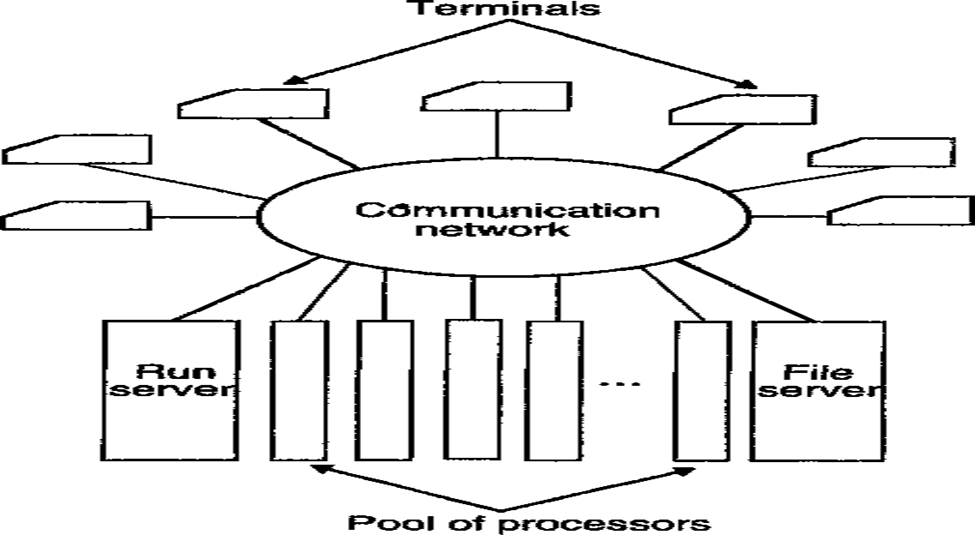
**Ưu điểm**:

- Tối ưu hóa sử dụng bộ xử lý bằng cách phân phối theo yêu cầu.

- Giảm thiểu lãng phí tài nguyên không sử dụng.

**Hạn chế**:

- Khó cài đặt và quản lý hệ thống phức tạp.



*Hình 2.4 Mô hình vùng các bộ xử lý*

### ***2.2.5 Mô hình kết hợp (Hybrid Model)***

Mô hình này kết hợp ưu điểm của mô hình trạm làm việc - máy phục vụ và mô hình vùng các bộ xử lý. Mô hình này phù hợp cho các môi trường có số lượng người dùng lớn, thực hiện các tác vụ đòi hỏi tính toán cao.

**Ưu điểm**:

- Khả năng mở rộng cao.

- Tối ưu hóa việc sử dụng tài nguyên hệ thống.

**Hạn chế**:

- Phức tạp hơn trong việc thiết kế và triển khai.

A diagram of a computer network

Description automatically generated

*Hình 2.5 Mô hình kết hợp*

**2.3. [Phần mềm hệ thống phổ biến giúp hình thành hệ thống phân tán.](#_heading=h.1mrcu09)**

***2.3.1 Hệ điều hành mạng (Network operating system)***

Hệ điều hành mạng (Network Operating System - NOS) là một loại hệ điều hành được thiết kế đặc biệt để hỗ trợ các chức năng liên quan đến mạng máy tính. NOS quản lý các tài nguyên mạng, cho phép các máy tính khác nhau trong cùng một mạng có thể giao tiếp, chia sẻ tài nguyên (như tập tin, máy in, kết nối Internet), và thực hiện các tác vụ mạng khác.

***2.3.2 Phần mềm trung gian (Middleware)***

Phần mềm trung gian (Middleware) là lớp phần mềm nằm giữa hệ điều hành và các ứng dụng, giúp kết nối, tương tác và trao đổi dữ liệu giữa các thành phần khác nhau của một hệ thống phần mềm, đặc biệt trong các hệ thống phân tán. Nó thường được sử dụng để quản lý các dịch vụ cơ bản cho các ứng dụng, như giao tiếp giữa các ứng dụng, quản lý phiên làm việc, và xử lý giao dịch.

***2.3.3 Hệ điều hành phân tán (Distributed operating system)***

Hệ điều hành phân tán (Distributed Operating System) là một loại hệ điều hành quản lý nhiều máy tính độc lập được kết nối qua mạng, nhưng hệ điều hành này khiến các tài nguyên của các máy tính đó hoạt động như thể chúng là một hệ thống duy nhất. Mục tiêu của hệ điều hành phân tán là cung cấp một môi trường tính toán liền mạch, nơi người dùng có thể tương tác với các tài nguyên và dịch vụ mà không cần biết chúng nằm trên máy tính nào trong mạng.

**2.4 Mục tiêu và những vấn đề cần giải quyết khi thiết kế hệ thống phân tán**

***2.4.1 Mục tiêu:***

*Tính trong suốt*

Hệ thống phân tán phải ẩn đi tính chất phân tán của nó, làm cho người dùng và ứng dụng không cần biết tài nguyên thực tế nằm ở đâu hoặc làm thế nào để truy cập chúng. Các loại tính trong suốt bao gồm:

* Minh bạch về vị trí: Người dùng không cần biết tài nguyên nằm trên máy nào.
* Minh bạch về di chuyển: Tài nguyên có thể di chuyển mà không ảnh hưởng đến người dùng.
* Minh bạch về lỗi: Người dùng không nhận ra lỗi khi một số thành phần của hệ thống gặp sự cố.

*Tính mở*

Hệ thống phân tán cần thiết kế theo các giao diện và giao thức mở để dễ dàng kết nối và tương tác với các hệ thống khác, có thể mở rộng và tương thích với các thành phần mới mà không cần thay đổi toàn bộ hệ thống.

*Tính tin cậy*

Đảm bảo hệ thống hoạt động ổn định và có khả năng khôi phục từ các lỗi phần cứng hoặc phần mềm, đảm bảo tính toàn vẹn dữ liệu và tính sẵn sàng của hệ thống ngay cả khi một số thành phần bị lỗi.

*Hiệu năng*

Hệ thống cần tối ưu hóa thời gian xử lý và đáp ứng các yêu cầu của người dùng một cách nhanh chóng, ngay cả khi số lượng yêu cầu lớn hoặc có sự tham gia của nhiều thành phần phân tán.

*Khả năng có thể mở rộng*

Hệ thống phải có khả năng mở rộng để xử lý lượng người dùng và dữ liệu tăng lên mà không làm giảm hiệu suất, cho phép thêm vào nhiều tài nguyên hơn mà không phải thay đổi kiến trúc cơ bản.

*Tính không đồng nhất của phần cứng*

Hệ thống phân tán phải có khả năng làm việc với nhiều loại phần cứng khác nhau, từ máy chủ mạnh đến các thiết bị di động yếu hơn, và các hệ điều hành khác nhau.

*Tính bảo mật*

Hệ thống cần đảm bảo an toàn cho dữ liệu và các giao dịch, bảo vệ khỏi các mối đe dọa như truy cập trái phép, xâm nhập hoặc tấn công. Bảo mật cần được thực hiện trên nhiều cấp độ, bao gồm việc mã hóa, xác thực người dùng và kiểm soát quyền truy cập.

*Tính tương thích với các hệ thống đã có*

Hệ thống phân tán cần phải tương thích với các hệ thống hiện có để tận dụng được các tài nguyên và dịch vụ có sẵn mà không phải thay thế hoàn toàn hạ tầng hiện có.

***2.4.2 Vấn đề cần giải quyết:***

*Vấn đề đặt tên*

Một hệ thống phân tán cần phải có cơ chế đặt tên nhất quán cho các tài nguyên (như máy tính, thiết bị, tệp tin, dịch vụ). Hệ thống cần phải đảm bảo rằng mọi tài nguyên có thể được nhận diện và truy cập thông qua một tên duy nhất và không gây ra xung đột.

*Cơ chế giao tiếp liên tiến trình*

Giao tiếp giữa các tiến trình trên các máy khác nhau trong hệ thống phân tán là một vấn đề phức tạp. Cần phải thiết kế các giao thức và cơ chế đảm bảo rằng các tiến trình có thể trao đổi dữ liệu một cách hiệu quả và an toàn qua mạng.

*Quản lý tài nguyên phân tán*

Việc quản lý tài nguyên trong hệ thống phân tán bao gồm việc phân bổ, điều phối và tối ưu hóa tài nguyên (CPU, bộ nhớ, lưu trữ) trên nhiều máy tính khác nhau để đảm bảo hiệu quả sử dụng và tránh hiện tượng quá tải hoặc xung đột tài nguyên.

*Vấn đề chịu đựng lỗi*

Hệ thống phân tán cần phải có khả năng chịu đựng lỗi (chẳng hạn như máy chủ bị hỏng hoặc mạng bị gián đoạn), đảm bảo rằng hệ thống vẫn hoạt động ngay cả khi có sự cố. Điều này yêu cầu các cơ chế phát hiện lỗi và khôi phục, cũng như khả năng sao lưu và dự phòng.

*Vấn đề bảo mật*

Bảo mật trong hệ thống phân tán cần được quan tâm ở mức cao, bởi vì các hệ thống này thường giao tiếp qua mạng, tạo cơ hội cho các cuộc tấn công mạng. Cần xây dựng các cơ chế bảo mật mạnh mẽ như mã hóa, xác thực người dùng, và kiểm soát truy cập để bảo vệ tài nguyên và dữ liệu trong hệ thống.

**2.5 [Các thư viện Celery, SCOOP và Pyro4](https://docs.google.com/document/d/1y7vkpKNMvsMfvPLDKfg7ER1W92-_KMUx/edit" \l "heading=h.3cqmetx)**

***2.5.1 Giới thiệu***

Ý tưởng cơ bản của **tính toán phân tán** là chia khối lượng công việc thành một số lượng tùy ý các nhiệm vụ, mỗi nhiệm vụ được chỉ định cho một máy tính trong mạng phân tán và hoàn thành chúng một cách chính xác. Cần đảm bảo các máy trong mạng luôn sẵn sàng (ví dụ như vấn đề độ trễ, sự cố không đoán trước của mạng hoặc máy tính). Do đó, cần một kiến trúc giám sát liên tục.

Vấn đề chính phát sinh từ việc sử dụng công nghệ này liên quan đến việc quản lý lưu lượng dữ liệu và quản lý các máy có hệ điều hành khác nhau, thường không tương thích với nhau. Mục tiêu của các mô hình này là cung cấp một khuôn khổ để mô tả sự hợp tác giữa các quá trình trong ứng dụng phân tán.

Mô hình phổ biến nhất là mô hình client-server, cho phép các quá trình trên các máy tính khác nhau hợp tác trong thời gian thực thông qua việc trao đổi thông điệp. Mô hình client-server thường được triển khai thông qua các lời gọi thủ tục từ xa (Remote Procedure Calls - RPC) hoặc qua mô hình đối tượng phân tán (Object-Oriented Middleware)

**2.6 Sử dụng Celery cho tác vụ phân tán**

Celery là một framework Python được sử dụng để quản lý các nhiệm vụ phân tán, theo cách tiếp phần mềm trung gian hướng đối tượng (Object-Oriented Middleware).

Tính năng chính: xử lý nhiều tác vụ nhỏ và phân phối đến một số lượng lớn các nút tính toán. Cuối cùng, kết quả của mỗi tác vụ sẽ được làm lại để tạo ra giải pháp tổng thể.

Các thành phần cơ bản: Module Celery và Message broker

***2.6.1 Message broker***

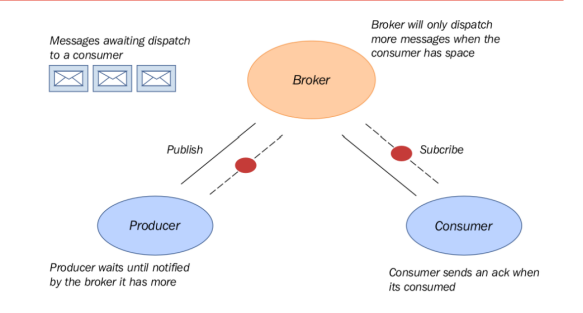
- Phần mềm trung gian - là một thành phần phần mềm độc lập với Celery.

- Được sử dụng để gửi và nhận tin nhắn cho các worker thực hiện nhiệm vụ phân tán ( trung gian trung chuyển message từ người gửi đến người nhận).

- Còn được gọi là phần mềm trung gian tin nhắn (Message từ middleware): liên quan đến việc trao đổi tin nhắn trong một mạng truyền thông.

- Kiểm tra, trung chuyển và điều hướng Message

- Celery hỗ trợ nhiều loại message brokers - trong dó đầy đủ nhất là RabbitMQ và Redis.

****

*Hình 2.6 Mô hình Celery*

Sơ đồ trong hình mô tả một mô hình giao tiếp giữa Producer (nhà sản xuất) và Consumer (người tiêu thụ) thông qua Broker (trung gian) trong một hệ thống phân tán, theo mô hình publish-subscribe (pub/sub).

***2.6.2 Producer (Nhà sản xuất)***

- Đây là nguồn gốc của các thông điệp. Producer tạo ra và gửi thông điệp đến Broker.

- Publish: Producer sẽ "đăng" (publish) thông điệp đến Broker. Sau khi đăng, Producer sẽ không tiếp tục gửi thêm cho đến khi nhận được thông báo từ Broker rằng có không gian trống để tiếp tục.

***2.6.3 Broker (Trung gian)***

- Broker đóng vai trò là trung gian giữa Producer và Consumer.

- Nhiệm vụ chính: Broker lưu trữ các thông điệp tạm thời và phân phối chúng tới Consumer.

- Broker sẽ chỉ gửi tiếp các thông điệp khi Consumer có không gian trống. Nếu không có không gian trống, Broker sẽ giữ lại các thông điệp cho đến khi Consumer sẵn sàng.

***2.6.4 Consumer (Người tiêu thụ)***

- Consumer đăng ký nhận các thông điệp từ Broker thông qua cơ chế "Subscribe" (đăng ký nhận).

- Sau khi nhận và xử lý xong thông điệp, Consumer sẽ gửi một thông báo "ack" (acknowledgement) đến Broker để xác nhận rằng thông điệp đã được xử lý thành công.

***2.6.5 Cơ chế chờ và thông báo***

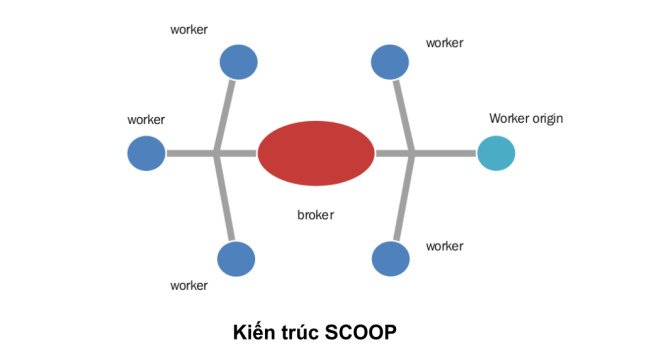
- Producer chờ: Khi Producer đã gửi đủ thông điệp và Broker chưa thể xử lý thêm (do Consumer chưa có không gian trống), Producer sẽ tạm ngưng gửi thêm và chờ thông báo từ Broker rằng có thể tiếp tục gửi.

- Consumer chờ và gửi xác nhận: Consumer sẽ chờ nhận thông điệp và sau khi xử lý xong sẽ gửi thông báo đến Broker.

**2.7 Tính toán khoa học với SCOOP**

**SCOOP (Scalable Concurrent Operations)**: Là một module Python để phân tán các tác vụ đồng thời (được gọi là Futures) trên các nút tính toán không đồng nhất.

Là một thư viện có đầy đủ chức năng được sử dụng chủ yếu trong các bài toán tính toán khoa học



*Hình 2.7 Mô hình SCOOP*

Dưới đây là giải thích các thành phần chính trong sơ đồ:

***2.7.1 Broker (Trung gian)***

- Đây là thành phần trung tâm điều phối công việc giữa các **Worker** (công nhân).

- Broker nhận các tác vụ từ "Worker origin" và phân phối chúng đến các **Worker** khác để xử lý.

***2.7.2 Worker origin***

- Đây là worker bắt đầu quá trình, tạo ra các nhiệm vụ và gửi chúng tới **Broker**.

- Sau đó, Broker sẽ điều phối các nhiệm vụ đó đến các Worker khác để xử lý.

**2.7.3 Worker (Công nhân)**

- Các Worker này là các nút trong hệ thống, thực hiện các tác vụ mà Broker phân phối.

- Mỗi worker hoạt động độc lập, nhận nhiệm vụ từ Broker và trả lại kết quả sau khi hoàn thành.

**2.7.4 Tương tác giữa các thành phần**

- Worker origin giao nhiệm vụ cho Broker.

- Broker sẽ điều phối nhiệm vụ này đến các Worker để thực hiện.

- Sau khi nhiệm vụ được hoàn thành, các Worker gửi lại kết quả cho Broker để tiếp tục xử lý.

**2.8 Gọi phương thức từ xa với pyro4**

Python Remote Objects (Pyro4) là một thư viện tương tự như Gọi phương thức từ xa - Remote Method Invocation (RMI) của Java.

Cho phép gọi một phương thức của một đối tượng từ xa (thuộc tiến trình khác và có khả năng nằm trên một máy khác) gần như là đối tượng là cục bộ (nghĩa là nó thuộc về cùng một tiến trình mà nó chạy lời gọi).

Cho phép xây dựng các ứng dụng trong đó các đối tượng có thể nói chuyện với nhau qua mạng mà không tốn nhiều công sức lập trình.

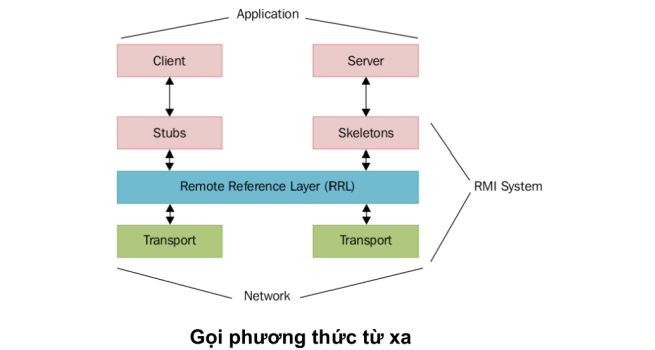
Cho phép mô hình hóa các tương tác giữa các tiến trình phân tán bằng cách sử dụng cùng một công cụ khái niệm (được sử dụng để biểu diễn các tương tác giữa các đối tượng khác nhau của một ứng dụng hoặc lệnh gọi phương thức).

Pyro4 cho phép quản lý và phân phối các đối tượng theo kiểu máy khách-máy chủ (Client – Server).

Các phần chính của hệ thống Pyro4 có thể chuyển đổi từ Client được gọi là đối tượng từ xa sang đối tượng được gọi để phục vụ chức năng.

Có thể sử dụng các lệnh gọi phương thức Python bình thường để gọi các đối tượng trên các máy khác.

Pyro là một thư viện Python thuần túy và chạy trên nhiều nền tảng và phiên bản Python khác nhau.



*Hình 2.8 Mô hình pyro4*

Hình ảnh này mô tả kiến trúc của hệ thống **RMI** (Remote Method Invocation), một cơ chế cho phép một chương trình trên máy khách gọi một phương thức trên máy chủ từ xa thông qua mạng.

***2.8.1 Client (Máy khách)***

Máy khách là ứng dụng yêu cầu gọi các phương thức từ xa trên máy chủ.

Tầng này giao tiếp với các lớp "Stubs" để thực hiện các lệnh gọi phương thức từ xa.

***2.8.2 Stubs***

Stubs là một đối tượng đại diện trên máy khách, đóng vai trò là "proxy" để giao tiếp với các phương thức từ xa.

Nó chuyển đổi các lệnh gọi phương thức cục bộ thành các yêu cầu mạng, gửi chúng đến máy chủ thông qua tầng RRL (Remote Reference Layer) và Transport.

***2.8.3 Remote Reference Layer (RRL)***

Tầng này quản lý các tham chiếu từ xa giữa máy khách và máy chủ.

Nó xử lý các thông tin tham chiếu đối tượng từ xa và chịu trách nhiệm quản lý quá trình gọi phương thức qua mạng.

***2.8.4 Transport (Tầng vận chuyển)***

Transport là tầng giao thức mạng, chịu trách nhiệm vận chuyển dữ liệu qua mạng giữa máy khách và máy chủ.

Nó có thể sử dụng các giao thức truyền tải như TCP/IP để đảm bảo dữ liệu đến nơi an toàn.

***2.8.5 Server (Máy chủ)***

Máy chủ là nơi chứa các đối tượng thực thi phương thức được gọi từ xa.

Khi nhận được yêu cầu từ máy khách thông qua Skeleton, nó sẽ thực thi phương thức và trả lại kết quả.

***2.8.6 Skeletons***

Skeletons là lớp đại diện trên máy chủ, nhận yêu cầu từ máy khách thông qua Stubs.

Nó sẽ chuyển yêu cầu tới đối tượng thực tế để thực thi phương thức và trả kết quả về cho Stubs qua mạng.

***2.8.7 Quy trình hoạt động***

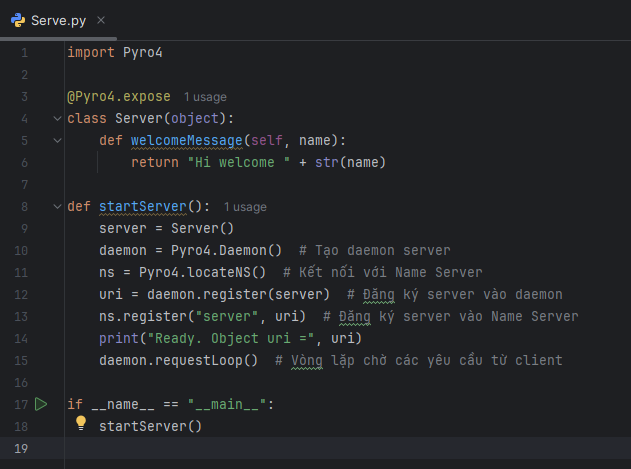
**Máy khách** gọi phương thức thông qua **Stub**. Stub sẽ đóng gói dữ liệu và gửi yêu cầu đến máy chủ thông qua tầng **Transport** và **RRL**.

**Skeleton** trên máy chủ nhận yêu cầu, giải mã, và gọi phương thức tương ứng trên đối tượng thực tế.

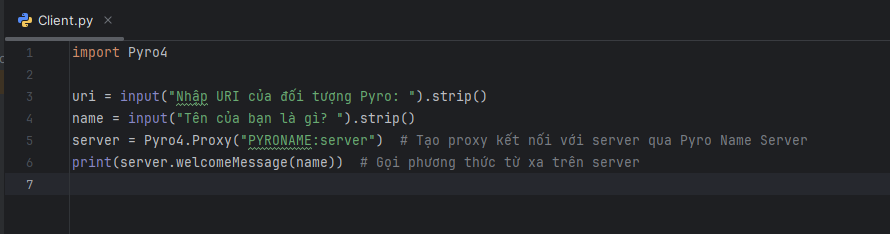
Sau khi phương thức hoàn thành, kết quả sẽ được chuyển ngược lại theo quy trình tương tự và trả về cho máy khách.

# CHƯƠNG III: [ỨNG DỤNG SỬ DỤNG PYRO4](#_heading=h.xvir7l)

Để hiểu rõ về việc sử dụng thư viện Pyro4, dưới đây là ví dụ về việc sử dụng Pyro4 để gọi phương thức của Server từ Client. Đầu tiên tiến hành cài đặt thư viện Pyro4 thông qua câu lệnh **pip install Pyro4,** ví dụ bao gồm hai file *Client.py* và *Serve.py.*

******

*Hình 3.1 Code file Serve.py*

******

*Hình 3.2 Code file Client.py*

## 3.1 Tổng quan về Pyro4:

**- Daemon**: Đây là một server được tạo ra để lắng nghe các yêu cầu từ client. Daemon chịu trách nhiệm quản lý các đối tượng từ xa và chuyển tiếp các yêu cầu từ client đến các đối tượng đó.

**- Proxy**: Đây là đại diện (proxy) của đối tượng server trên client. Client sẽ tương tác với proxy như thể nó đang làm việc trực tiếp với đối tượng gốc.

**- Name Server**: Đây là một loại “sổ địa chỉ” của các đối tượng từ xa, giúp client tìm kiếm các đối tượng trên mạng bằng tên thay vì phải biết địa chỉ mạng cụ thể.

## 3.2 Cách hoạt động:

### ***3.2.1 File Serve.py***

Server sẽ cung cấp một phương thức từ xa mà client có thể gọi. Khi khởi động, server tạo ra một daemon và đăng ký đối tượng của nó vào Pyro Name Server. Daemon này sẽ lắng nghe yêu cầu từ client và chuyển nó tới đối tượng server.

**Khởi tạo đối tượng Server**:  
 *server = Server()*

Đây là nơi tạo ra đối tượng Server, đối tượng này sẽ cung cấp phương thức welcomeMessage.

**Tạo daemon**:  
 *daemon = Pyro4.Daemon()*

Daemon là nơi mà đối tượng sẽ "sống", nó sẽ lắng nghe các yêu cầu từ xa và chuyển chúng đến đối tượng tương ứng.

**Kết nối với Name Server**:  
 *ns = Pyro4.locateNS()*

Lệnh này giúp server kết nối với **Pyro Name Server**, nơi lưu trữ và quản lý các đối tượng được chia sẻ qua mạng.

**Đăng ký đối tượng vào daemon**:  
 *uri = daemon.register(server)*

Đây là bước đăng ký đối tượng **server** vào daemon. Daemon sẽ biết rằng nó phải chuyển các yêu cầu từ client đến đối tượng này.

**Đăng ký đối tượng vào Name Server**:  
 *ns.register("server", uri)*

Bước này đăng ký đối tượng server với tên “**server**” vào Name Server. Client sẽ sử dụng tên này để tìm và gọi đến server.

**Vòng lặp yêu cầu**:  
 *daemon.requestLoop()*

Đây là vòng lặp chính của server, cho phép server liên tục lắng nghe và xử lý các yêu cầu từ client.

### ***3.2.2 File Client.py***

Client tạo một proxy để kết nối với đối tượng server thông qua tên đã đăng ký trên Name Server, sau đó gọi phương thức từ xa welcomeMessage.

**Nhập tên của server từ Name Server**:  
 *server = Pyro4.Proxy("PYRONAME:server")*

Dòng này sử dụng Proxy để tạo một đối tượng đại diện (proxy) cho server mà Name Server quản lý. Proxy sẽ "đóng vai" đối tượng server, và tất cả các lệnh client gửi qua proxy sẽ được chuyển tiếp đến server thực tế.

**Gọi phương thức từ xa**:  
 *print(server.welcomeMessage(name))*

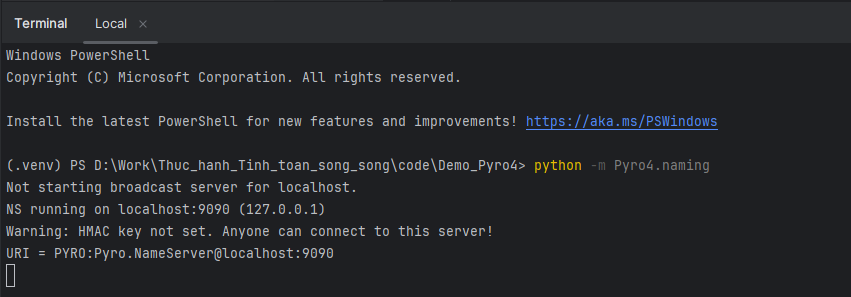
Client sẽ gọi phương thức welcomeMessage thông qua proxy. Kết quả của phương thức này sẽ được trả về từ server và hiển thị trên client.

## 3.3. Name Server

**Khởi động Name Server**:  
 *python -m Pyro4.naming*

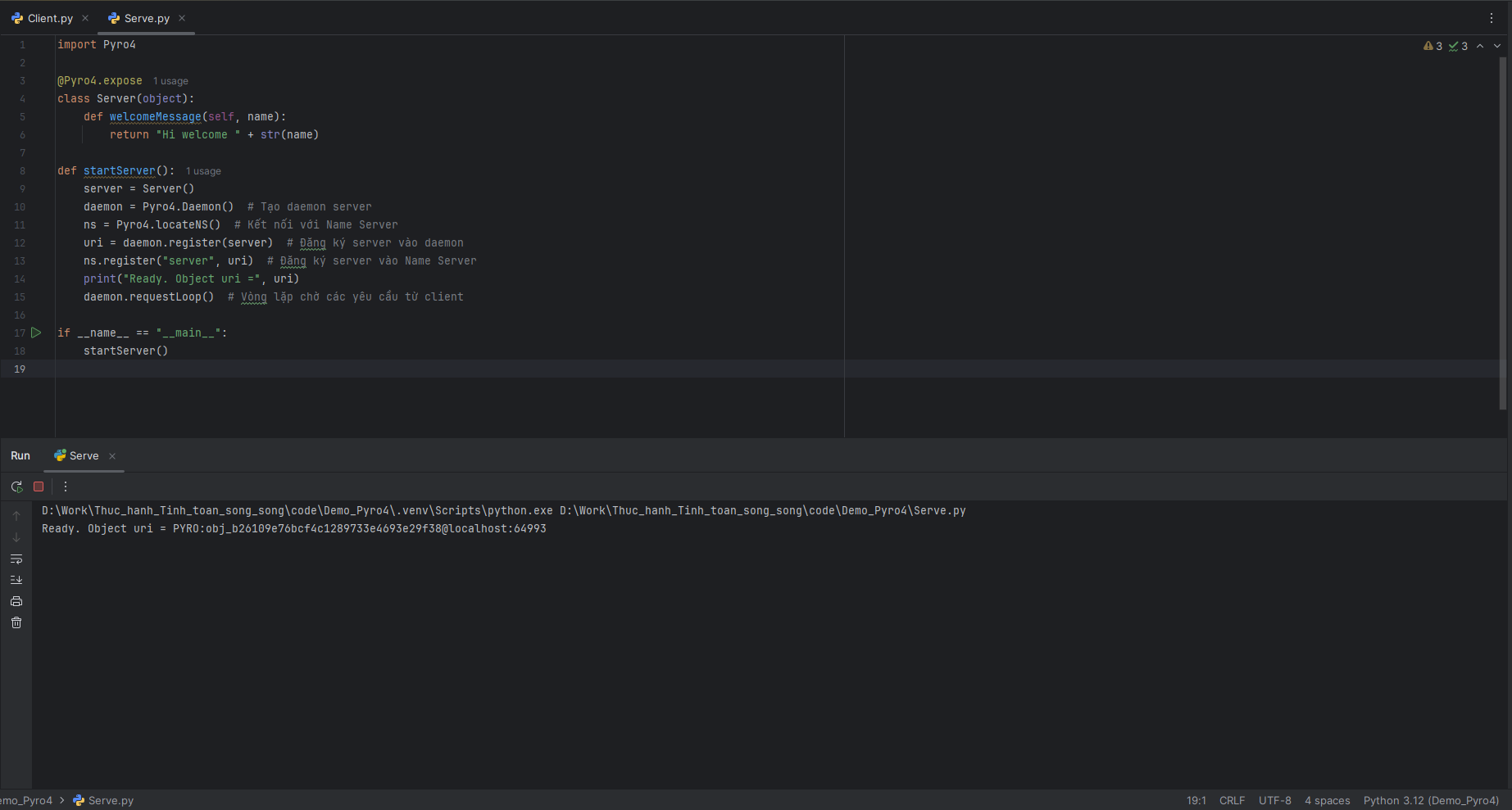
Name Server giúp quản lý các đối tượng từ xa. Khi server khởi động, nó sẽ đăng ký đối tượng của mình với một tên trong Name Server. Client sau đó chỉ cần biết tên này để tìm và kết nối với server.

**Kết quả chương trình:**

****

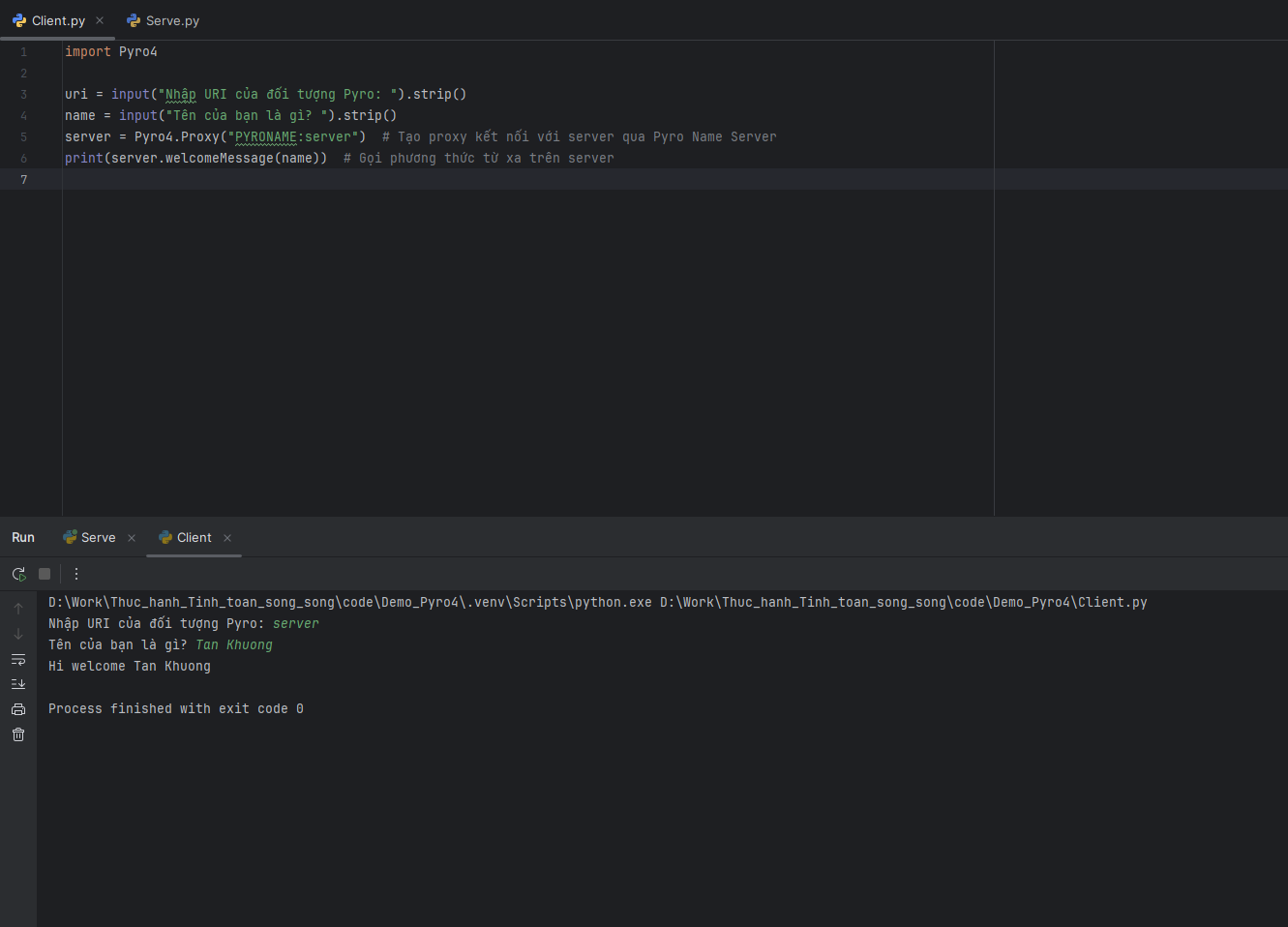
*Hình 3.3 Khởi tạo Server*

**Serve.py**

****

*Hình 3.4 Kết quả trả về của Server*

**Client.py**

****

*Hình 3.5 Kết quả trả về của Client*

**Cơ chế hoạt động**

* **Server** đăng ký đối tượng của nó vào **Name Server**.
* **Client** sử dụng **Proxy** để tương tác với đối tượng trên **Server**.
* **Pyro4** lo liệu tất cả các chi tiết kỹ thuật liên quan đến mạng, giúp việc gọi các phương thức từ xa trở nên dễ dàng như gọi phương thức cục bộ.

Hệ thống này rất hữu ích trong các ứng dụng phân tán, nơi bạn cần các đối tượng hoạt động từ xa nhưng lại muốn giữ sự đơn giản trong việc gọi phương thức.

# CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 4.1. Kết luận

Tính toán phân tán đã trở thành một giải pháp thiết yếu trong việc giải quyết các bài toán lớn, phức tạp và đòi hỏi tài nguyên vượt quá khả năng của một máy tính đơn lẻ. Thông qua việc chia nhỏ bài toán và phân phối các tác vụ đến nhiều máy tính hoặc nút mạng khác nhau, hệ thống phân tán đã mở ra cơ hội tăng cường **hiệu suất xử lý**, **nâng cao khả năng chịu lỗi**, và **tối ưu hóa tài nguyên**.

Trong quá trình nghiên cứu, chúng ta đã tìm hiểu các **mô hình hệ thống phân tán**, bao gồm máy tính tầm trung, trạm làm việc, và vùng các bộ xử lý, cũng như cách các mô hình này được sử dụng trong các môi trường cụ thể. Báo cáo cũng làm rõ vai trò của các công cụ hỗ trợ như **Celery**, **SCOOP**, và **Pyro4**, cho thấy chúng là những công cụ mạnh mẽ trong việc triển khai và quản lý hệ thống phân tán.

Bên cạnh các ưu điểm nổi bật, việc triển khai tính toán phân tán cũng đặt ra nhiều thách thức, bao gồm **bảo mật**, **quản lý tài nguyên**, và **đồng bộ dữ liệu**. Những thách thức này yêu cầu các giải pháp kỹ thuật tối ưu và cơ chế quản lý hiệu quả để đảm bảo tính toàn vẹn và hiệu năng của hệ thống.

Nhìn chung, tính toán phân tán không chỉ là một hướng tiếp cận lý thuyết mà còn có ứng dụng thực tiễn rộng rãi, đặc biệt trong các lĩnh vực như **trí tuệ nhân tạo**, **khoa học dữ liệu**, và **mô phỏng vật lý**. Các doanh nghiệp và tổ chức đang ngày càng tận dụng tính toán phân tán để triển khai các hệ thống thông minh và hiệu quả, phục vụ nhu cầu xử lý dữ liệu lớn và tăng trưởng liên tục của họ.

## 4.2. Hướng phát triển

Để tính toán phân tán tiếp tục phát triển và đáp ứng các yêu cầu ngày càng cao của công nghệ hiện đại, cần tập trung vào những hướng phát triển sau:

### **4.2.1. Tích hợp với các công nghệ mới**

**- Điện toán đám mây (Cloud Computing):** Phát triển hệ thống phân tán trên các nền tảng đám mây giúp tận dụng tài nguyên linh hoạt và mở rộng quy mô dễ dàng.

**- Điện toán biên (Edge Computing):** Phân phối tính toán gần người dùng hơn để giảm độ trễ và tăng hiệu năng, đặc biệt trong các ứng dụng IoT.

**- Trí tuệ nhân tạo (AI) và Máy học:** Tính toán phân tán giúp tăng tốc quá trình huấn luyện mô hình AI trên dữ liệu lớn.

### **4.2.2. Tăng cường bảo mật và quản lý tài nguyên**

**- Bảo mật trong môi trường phân tán:** Phát triển các giao thức mã hóa và xác thực mạnh mẽ để bảo vệ dữ liệu trong quá trình truyền tải.

**- Quản lý tài nguyên thông minh:** Sử dụng các thuật toán tối ưu hóa động để tự động phân bổ tài nguyên, giảm thiểu tình trạng quá tải hoặc lãng phí.

### **4.2.3. Phát triển các công cụ hỗ trợ mới**

**- Framework mới cho lập trình phân tán:** Các công cụ dễ sử dụng và hiệu quả hơn sẽ giúp lập trình viên triển khai hệ thống phân tán nhanh chóng và ít lỗi hơn.

**- Cải thiện Celery và các message broker:** Phát triển thêm các tính năng để hỗ trợ khả năng chịu lỗi và tối ưu hóa tốc độ xử lý cho hệ thống tác vụ lớn.

### **4.2.4. Ứng dụng trong các lĩnh vực mới**

**- Khoa học dữ liệu và phân tích dữ liệu lớn:** Sử dụng tính toán phân tán để phân tích dữ liệu nhanh hơn và đưa ra quyết định kịp thời.

**- Mô phỏng vật lý và y học:** Mở rộng tính toán phân tán trong các nghiên cứu y tế và mô phỏng phức tạp như dự báo dịch bệnh hoặc thiết kế thuốc.

### **4.2.5. Hướng tới các hệ thống phi tập trung và blockchain**

Sự kết hợp giữa tính toán phân tán và **blockchain** có thể tạo ra các hệ thống phi tập trung, tăng tính bảo mật và minh bạch trong các giao dịch dữ liệu. Điều này đặc biệt hữu ích trong lĩnh vực **tài chính**, **chuỗi cung ứng**, và **quản lý dữ liệu y tế**.

## 4.3. Kết luận và Tầm nhìn tương lai

Tính toán phân tán sẽ tiếp tục là một xu hướng quan trọng trong thời đại **chuyển đổi số**. Các tổ chức và doanh nghiệp không chỉ cần khai thác hiệu quả các tài nguyên tính toán hiện có mà còn phải thích ứng với những công nghệ mới để duy trì tính cạnh tranh. Trong tương lai, các hệ thống phân tán sẽ ngày càng thông minh hơn nhờ vào sự kết hợp với AI, dữ liệu lớn, và các mô hình điện toán tiên tiến khác.

Những nỗ lực nghiên cứu và phát triển không ngừng trong lĩnh vực này sẽ tiếp tục mang lại những cải tiến đột phá, giúp con người giải quyết những vấn đề phức tạp hơn và tối ưu hóa hệ thống trong mọi lĩnh vực của cuộc sống.

**CHƯƠNG V: TÀI LIỆU THAM KHẢO**

1. **Andrew S. Tanenbaum & Maarten Van Steen** (2017). *Distributed Systems: Principles and Paradigms*. Pearson Education.

Cuốn sách cung cấp cơ sở lý thuyết về hệ thống phân tán, các mô hình và giao thức liên quan.

1. **Mullender, S., et al.** (1990). *Amoeba: A Distributed Operating System for the 1990s*. IEEE Computer Society.

Bài viết trình bày hệ điều hành Amoeba, một trong những mô hình vùng bộ xử lý tiêu biểu cho hệ thống phân tán.

1. **Cheriton, D.R.** (1988). *The V-System: A Distributed Operating System*. ACM Transactions on Computer Systems.

Bài báo giới thiệu mô hình hệ thống trạm làm việc - máy phục vụ với cơ chế phân phối tài nguyên hiệu quả.

1. **Celery Project Documentation**. *Celery: Distributed Task Queue Framework*.

Truy cập tại: <https://docs.celeryproject.org>

Tài liệu hướng dẫn chi tiết về cách sử dụng Celery trong quản lý tác vụ phân tán.

1. **SCOOP Project Documentation**. *Scalable Concurrent Operations in Python*.

Truy cập tại:<https://github.com/soravux/scoop>

Thông tin về cách sử dụng SCOOP trong các ứng dụng tính toán khoa học.

1. **Pyro4 Project Documentation**. *Python Remote Objects Library*.

Truy cập tại:<https://github.com/irmen/Pyro4>

Tài liệu về Pyro4, thư viện hỗ trợ gọi phương thức từ xa trong các hệ thống client-server.

1. **George Coulouris et al.** (2012). *Distributed Systems: Concepts and Design*. Pearson.

Cuốn sách cung cấp nền tảng về các khái niệm và thiết kế hệ thống phân tán.