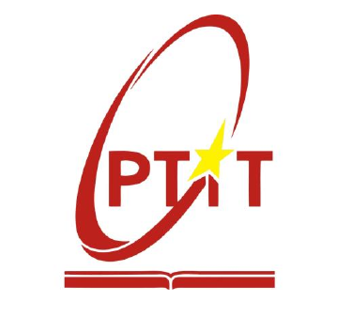
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

----🙡🕮🙣----



**BÀI TIỂU LUẬN MÔN THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI:**

**HỆ THỐNG TMĐT SHOPGRIDS PHÂN TÁN**

|  |  |
| --- | --- |
| **Giảng viên** | **:PGS.TS. PHẠM VĂN CƯỜNG** |
| **Họp viên thực hiện** | **: HOÀNG ĐỨC CƯỜNG - B24CHHT059** |
|  | **NGUYỄN ĐỨC THỨC - B24CHHT094**  **PHẠM ĐỨC LONG - B24CHHT083** |
| **Lớp** | **:M24CQHT02-B** |

**1. MỞ ĐẦU**

**1.1. Bối cảnh và lý do nghiên cứu**

Trong kỷ nguyên Cách mạng Công nghiệp 4.0, sự hội tụ của các công nghệ nền tảng như điện toán đám mây (Cloud Computing), trí tuệ nhân tạo (Artificial Intelligence), Internet vạn vật (Internet of Things – IoT) và chuỗi khối (Blockchain) đã làm thay đổi căn bản phương thức vận hành và cung ứng dịch vụ của các doanh nghiệp. Trong đó, thương mại điện tử (TMĐT) đã trở thành một trong những lĩnh vực tiên phong trong việc ứng dụng công nghệ nhằm nâng cao trải nghiệm khách hàng, tối ưu hóa chuỗi cung ứng và mở rộng thị trường toàn cầu.

Sự gia tăng đột biến về lưu lượng truy cập, khối lượng giao dịch và yêu cầu trải nghiệm cá nhân hóa đặt ra những thách thức lớn cho các hệ thống TMĐT. Các hệ thống monolithic truyền thống bộc lộ hạn chế về khả năng mở rộng, tính sẵn sàng và độ tin cậy khi phải đáp ứng nhu cầu của hàng triệu người dùng đồng thời. Do đó, hệ thống phân tán (Distributed System) – với khả năng xử lý song song, phân bổ tài nguyên động và đảm bảo tính sẵn sàng cao – trở thành hướng tiếp cận tất yếu.

Trong bối cảnh này, ShopGrids được định hướng phát triển như một nền tảng TMĐT phân tán chuyên biệt cho thiết bị điện tử, đáp ứng các yêu cầu: khả năng mở rộng linh hoạt, hiệu năng cao, an toàn thông tin và dễ dàng tích hợp công nghệ mới. Việc nghiên cứu và thiết kế ShopGrids dưới góc độ hệ thống phân tán không chỉ mang giá trị thực tiễn cho doanh nghiệp mà còn đóng góp về mặt học thuật trong việc ứng dụng mô hình microservices và kiến trúc cloud-native cho TMĐT.

**1.2. Vấn đề và khoảng trống nghiên cứu**

Các nghiên cứu và triển khai hệ thống TMĐT hiện nay tập trung nhiều vào chức năng nghiệp vụ (quản lý sản phẩm, đơn hàng, thanh toán, vận chuyển) nhưng ít chú trọng đến việc tích hợp kiến trúc phân tán ngay từ giai đoạn thiết kế. Điều này dẫn đến các vấn đề:

* Khó mở rộng quy mô khi số lượng người dùng tăng nhanh.
* Điểm nghẽn xử lý tại các thành phần tập trung.
* Khó triển khai phân tán trên môi trường đa đám mây.

Khoảng trống này mở ra nhu cầu nghiên cứu và triển khai các hệ thống TMĐT được kiến trúc hóa theo hướng phân tán ngay từ đầu, để vừa đảm bảo tính năng nghiệp vụ vừa tối ưu hiệu năng, độ tin cậy và bảo mật.

**1.3. Mục tiêu nghiên cứu**

Nghiên cứu này hướng tới:

1. Phân tích cơ sở lý thuyết và đặc trưng của hệ thống phân tán, liên hệ tới nhu cầu và đặc điểm của TMĐT.
2. Đề xuất mô hình kiến trúc phân tán cho ShopGrids dựa trên microservices và cloud-native.
3. Thiết kế chi tiết các thành phần hệ thống, cơ chế giao tiếp, bảo mật và đồng bộ dữ liệu trong môi trường phân tán.
4. Đánh giá khả năng mở rộng, hiệu năng và độ tin cậy của mô hình đề xuất.

**1.4. Đối tượng và phạm vi nghiên cứu**

Đối tượng nghiên cứu: Hệ thống ShopGrids và các mô hình kiến trúc phân tán trong TMĐT.

Phạm vi nghiên cứu: Tập trung vào khía cạnh kiến trúc và kỹ thuật của hệ thống phân tán, không đi sâu vào chi tiết vận hành marketing hay logistics. Môi trường triển khai giả định trên nền tảng cloud public (AWS, GCP, Azure) và container orchestration (Kubernetes).

**1.5. Ý nghĩa khoa học và thực tiễn**

Ý nghĩa khoa học: Góp phần bổ sung nghiên cứu về việc áp dụng các nguyên lý của hệ thống phân tán trong thiết kế nền tảng TMĐT, kết hợp giữa microservices, cloud-native và các công nghệ hỗ trợ như message broker, caching layer, API gateway.

Ý nghĩa thực tiễn: Cung cấp một khung kiến trúc mẫu có khả năng triển khai thực tế, đáp ứng các yêu cầu vận hành của doanh nghiệp vừa và nhỏ, đồng thời dễ dàng mở rộng cho quy mô lớn.

**2. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN**

**2.1. Khái niệm hệ thống phân tán**

Theo định nghĩa của Tanenbaum & Van Steen (2017), *hệ thống phân tán* (Distributed System) là “tập hợp các máy tính độc lập, được người dùng nhìn nhận như một hệ thống thống nhất, phối hợp với nhau thông qua việc trao đổi thông điệp để đạt được một mục tiêu chung”.

Các hệ thống này không bị ràng buộc bởi vị trí địa lý, có thể bao gồm các máy chủ, dịch vụ và thiết bị ở nhiều khu vực khác nhau, kết nối thông qua mạng Internet hoặc mạng chuyên dụng. Hệ thống phân tán có mặt trong nhiều lĩnh vực như TMĐT, mạng xã hội, xử lý dữ liệu lớn, dịch vụ tài chính và hệ thống IoT.

**2.2. Đặc trưng của hệ thống phân tán**

Một hệ thống phân tán thường thể hiện các đặc trưng cốt lõi:

1. Tính minh bạch (Transparency):
   * *Vị trí (Location transparency):* Người dùng không cần biết tài nguyên nằm ở đâu.
   * *Sao chép (Replication transparency):* Nhiều bản sao dữ liệu tồn tại mà không ảnh hưởng đến truy cập.
   * *Lỗi (Failure transparency):* Hệ thống vẫn hoạt động khi một số thành phần gặp sự cố.
2. Khả năng mở rộng (Scalability): Có thể mở rộng về kích thước (thêm node), địa lý (mở rộng vùng triển khai) và hành chính (quản trị nhiều tổ chức khác nhau).
3. Tính chịu lỗi (Fault Tolerance): Khả năng tiếp tục cung cấp dịch vụ khi một phần hệ thống hỏng.
4. Đồng bộ dữ liệu (Data Consistency): Đảm bảo tính nhất quán theo các mô hình như *Strong Consistency*, *Eventual Consistency* hoặc *Causal Consistency*.

**2.3. So sánh hệ thống tập trung và phân tán**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Hệ thống tập trung** | **Hệ thống phân tán** |
| Kiến trúc | Một máy chủ chính | Nhiều node liên kết |
| Khả năng mở rộng | Hạn chế | Cao (thêm node dễ dàng) |
| Tính sẵn sàng | Thấp (Single Point of Failure) | Cao nhờ nhân bản và dự phòng |
| Chi phí ban đầu | Thấp hơn | Cao hơn |
| Quản lý | Đơn giản | Phức tạp hơn |

**2.4. Mô hình kiến trúc hệ thống phân tán**

1. Client–Server: Phổ biến, client gửi yêu cầu, server xử lý và phản hồi.
2. Peer-to-Peer (P2P): Các node bình đẳng, vừa đóng vai trò client vừa làm server.
3. Microservices: Chia hệ thống thành các dịch vụ nhỏ, độc lập, giao tiếp qua API hoặc message broker.
4. Event-driven Architecture: Các thành phần phản ứng theo sự kiện, thích hợp cho xử lý bất đồng bộ.

**2.5. Giao tiếp trong hệ thống phân tán**

* Giao thức đồng bộ: RESTful API qua HTTP/HTTPS, gRPC.
* Giao thức bất đồng bộ: Message Queue (RabbitMQ, Kafka), Pub/Sub (Google Pub/Sub, AWS SNS).
* Quản lý dịch vụ: Service Discovery (Eureka, Consul), API Gateway (Kong, NGINX, AWS API Gateway).

**2.6. Bảo mật trong hệ thống phân tán**

Các thách thức bảo mật bao gồm xác thực, phân quyền, mã hóa dữ liệu, chống tấn công DDoS và bảo vệ API. Một số giải pháp:

* Xác thực & phân quyền: OAuth2, JWT, Multi-Factor Authentication (MFA).
* Mã hóa: TLS/SSL cho truyền dữ liệu, AES-256 cho lưu trữ.
* Tường lửa ứng dụng web (WAF): Chặn tấn công SQL Injection, XSS.

**2.7. Ứng dụng của hệ thống phân tán trong thương mại điện tử**

Trong TMĐT, hệ thống phân tán cho phép:

* Xử lý đồng thời lượng lớn yêu cầu mua hàng, thanh toán và truy vấn sản phẩm.
* Phân phối nội dung tĩnh qua CDN tới người dùng ở nhiều khu vực.
* Kết hợp các dịch vụ AI/ML cho gợi ý sản phẩm, phân tích hành vi.
* Tích hợp hệ thống thanh toán và logistics qua API.

**3. KIẾN TRÚC PHÂN TÁN CỦA SHOPGRIDS**

**3.1. Định hướng kiến trúc**

ShopGrids được thiết kế theo kiến trúc microservices kết hợp với nguyên tắc cloud-native nhằm đảm bảo khả năng mở rộng (scalability), tính sẵn sàng (availability) và khả năng phục hồi (resilience). Mỗi thành phần nghiệp vụ chính được triển khai như một dịch vụ độc lập, có thể phát triển, triển khai và nâng cấp mà không ảnh hưởng đến các thành phần khác.

Các nguyên tắc cơ bản:

* Tách biệt dịch vụ (Service Decoupling): Mỗi dịch vụ xử lý một chức năng duy nhất, giảm độ phụ thuộc.
* Cơ sở dữ liệu riêng (Database-per-service): Mỗi dịch vụ quản lý dữ liệu của mình, đảm bảo tính độc lập và giảm xung đột truy vấn.
* Giao tiếp qua API hoặc Message Broker: Tận dụng REST/GraphQL cho truy vấn đồng bộ, RabbitMQ/Kafka cho xử lý bất đồng bộ.
* Tự động triển khai và mở rộng: Thông qua Docker và Kubernetes.

**3.2. Thành phần kiến trúc**

**1. Lớp giao tiếp với người dùng (Frontend Layer)**

* Web App: Phát triển bằng React/Vue.js, tích hợp API Gateway.
* Mobile App: Ứng dụng native/hybrid sử dụng API REST/GraphQL từ backend.
* Tích hợp CDN: Đảm bảo tốc độ tải nhanh trên toàn cầu.

**2. API Gateway**

* Điều phối các yêu cầu từ client đến đúng microservice.
* Cung cấp xác thực, phân quyền, hạn chế tốc độ (rate limiting) và ghi log truy cập.

**3. Các microservice cốt lõi**

* User Service: Quản lý thông tin người dùng, vai trò, xác thực, bảo mật (MFA, JWT/OAuth2).
* Product Service: Quản lý danh mục sản phẩm, tồn kho, giá.
* Order Service: Xử lý đơn hàng, trạng thái, tích hợp thanh toán.
* Payment Service: Kết nối cổng thanh toán (VNPay, PayPal, Stripe).
* Recommendation Service: Gợi ý sản phẩm dựa trên AI/ML.

**4. Message Broker**

* Sử dụng RabbitMQ/Kafka cho xử lý sự kiện (ví dụ: khi đơn hàng được xác nhận, gửi thông báo và cập nhật tồn kho).

**5. Lớp cơ sở dữ liệu phân tán**

* Mỗi microservice quản lý cơ sở dữ liệu riêng: MySQL/PostgreSQL cho dữ liệu quan hệ, MongoDB/ElasticSearch cho dữ liệu phi cấu trúc.
* Redis/Memcached làm caching layer để tăng tốc độ phản hồi.

**6. Hạ tầng triển khai**

* Containerization: Mỗi microservice đóng gói trong Docker container.
* Orchestration: Kubernetes quản lý việc mở rộng, cân bằng tải, phục hồi khi lỗi.
* Monitoring & Logging: Prometheus và Grafana theo dõi hiệu năng; ELK Stack (Elasticsearch, Logstash, Kibana) phân tích log.

**3.3. Sơ đồ kiến trúc tổng thể**

sql

Sao chépChỉnh sửa

+----------------+ +----------------+

| Web/Mobile |<----->| API Gateway |

+----------------+ +--------+-------+

|

-------------------------------------------------

| | | | |

+---------+--+ +------+-------+ +--+-------+ +--+------+----+

| User Svc | | Product Svc | | Order Svc | | Payment Svc |

+-----+----+ +------+-----+ +----+------+ +-------+-------+

| | | |

| | | |

+---+---+ +---+---+ +---+---+ +---+---+

| MySQL | | MySQL | | MySQL | | MySQL |

+-------+ +-------+ +-------+ +-------+

<< Event Bus / Message Broker (Kafka/RabbitMQ) >>

| | |

+----------------+------------------+

**3.4. Cơ chế hoạt động**

1. Người dùng gửi yêu cầu qua web/mobile → API Gateway.
2. API Gateway xác thực (JWT/OAuth2), phân quyền và định tuyến đến microservice thích hợp.
3. Microservice xử lý nghiệp vụ:
   * Nếu là tác vụ đồng bộ (truy vấn sản phẩm, đăng nhập): dùng REST/GraphQL.
   * Nếu là tác vụ bất đồng bộ (thanh toán, gợi ý sản phẩm, gửi email): đưa vào hàng đợi của RabbitMQ/Kafka.
4. Cơ sở dữ liệu riêng của từng microservice đảm bảo tính nhất quán nội bộ; các dịch vụ đồng bộ hóa dữ liệu qua sự kiện.
5. Hệ thống giám sát phát hiện lỗi và tự động khởi động lại container khi cần.

**3.5. Ưu điểm của kiến trúc này đối với ShopGrids**

* Mở rộng theo chiều ngang: Dễ dàng tăng số lượng instance cho từng dịch vụ.
* Tính sẵn sàng cao: Không có single point of failure; lỗi ở một dịch vụ không làm sập toàn hệ thống.
* Dễ bảo trì và phát triển: Mỗi nhóm phát triển có thể làm việc độc lập trên một microservice.
* Khả năng tích hợp: Dễ kết nối dịch vụ ngoài (thanh toán, vận chuyển, AI).

**CHƯƠNG 4**

**BẢO MẬT VÀ HIỆU NĂNG TRONG HỆ THỐNG PHÂN TÁN SHOPGRIDS**

**4.1. Thách thức bảo mật trong hệ thống phân tán**

Hệ thống phân tán, đặc biệt trong TMĐT, có bề mặt tấn công rộng hơn so với hệ thống tập trung, do tồn tại nhiều điểm truy cập và giao tiếp giữa các dịch vụ. Một số thách thức chính:

1. Bảo mật liên miền (Cross-domain security): Các dịch vụ hoạt động ở nhiều vùng địa lý và miền khác nhau, dễ bị tấn công khi truyền dữ liệu qua Internet.
2. Nguy cơ tấn công vào API: Các endpoint REST/GraphQL có thể bị khai thác nếu không kiểm soát quyền truy cập.
3. Quản lý danh tính và quyền truy cập (IAM): Khó duy trì đồng bộ thông tin xác thực giữa nhiều dịch vụ.
4. Bảo vệ dữ liệu nhạy cảm: Thông tin cá nhân, dữ liệu thanh toán cần tuân thủ các tiêu chuẩn như PCI DSS, GDPR.

**4.2. Chiến lược bảo mật đề xuất cho ShopGrids**

Để đáp ứng yêu cầu bảo mật, ShopGrids áp dụng mô hình phòng thủ nhiều lớp (Defense in Depth) và tuân thủ ISO/IEC 27001 cùng khuyến nghị OWASP Top 10.

1. Bảo mật truy cập và xác thực

* Áp dụng OAuth2 và JWT để quản lý phiên làm việc cho API.
* Multi-Factor Authentication (MFA) cho tài khoản quản trị.
* Triển khai API Gateway với cơ chế Rate Limiting và IP Whitelisting.

**2. Bảo mật dữ liệu**

* Mã hóa dữ liệu khi truyền (TLS 1.3) và khi lưu trữ (AES-256).
* Hash mật khẩu bằng bcrypt/scrypt với salt ngẫu nhiên.
* Phân vùng dữ liệu theo mức độ nhạy cảm, hạn chế quyền truy cập.

**3. Giám sát và phát hiện xâm nhập**

Triển khai IDS/IPS (Intrusion Detection/Prevention Systems) như Wazuh/Snort.

Ghi log tập trung (Centralized Logging) qua ELK Stack để phân tích hành vi bất thường.

**4. Quản lý cấu hình và triển khai**

Quản lý bí mật (secrets) với HashiCorp Vault hoặc AWS Secrets Manager.

Áp dụng CI/CD pipeline với kiểm tra bảo mật (SAST, DAST) tự động.

**4.3. Thách thức hiệu năng trong hệ thống phân tán**

Hiệu năng là yếu tố sống còn với TMĐT, đặc biệt trong các dịp cao điểm (sale 11.11, Black Friday). Các vấn đề thường gặp:

1. Độ trễ mạng (Network latency) giữa các dịch vụ.
2. Nghẽn cổ chai tại cơ sở dữ liệu hoặc API Gateway.
3. Xử lý đồng thời khi lượng request tăng đột biến.
4. Chi phí tài nguyên tăng nhanh khi mở rộng hệ thống.

**4.4. Kỹ thuật tối ưu hiệu năng cho ShopGrids**

**1. Caching và giảm tải cơ sở dữ liệu**

* Redis/Memcached để lưu trữ tạm kết quả truy vấn phổ biến.
* Áp dụng *Write-through* hoặc *Read-through cache* để đồng bộ dữ liệu.

**2. Tối ưu giao tiếp dịch vụ**

* Sử dụng gRPC cho truyền dữ liệu nhị phân tốc độ cao giữa các microservice.
* Gộp (batch) yêu cầu thay vì gửi nhiều request nhỏ lẻ.

**3. Tự động mở rộng (Autoscaling)**

* Kubernetes Horizontal Pod Autoscaler (HPA) dựa trên CPU, RAM, hoặc lượng request.
* Kết hợp Vertical Pod Autoscaler (VPA) cho dịch vụ quan trọng.

**4. Phân phối nội dung và cân bằng tải**

* Sử dụng CDN (Cloudflare, AWS CloudFront) cho tài nguyên tĩnh.
* Triển khai Load Balancer (Nginx, HAProxy) phân phối đều request.

**5. Giám sát hiệu năng**

* Prometheus thu thập số liệu, Grafana hiển thị dashboard.
* Thiết lập cảnh báo (Alertmanager) khi vượt ngưỡng cho phép.

**4.5. Liên hệ tới ShopGrids**

Với kiến trúc microservices và triển khai trên cloud, ShopGrids có thể:

* Đảm bảo thời gian phản hồi <200ms cho 95% yêu cầu trong điều kiện tải cao.
* Duy trì SLA (Service Level Agreement) ở mức ≥99.9%.
* Chịu được các đợt tăng tải gấp 5–10 lần nhờ autoscaling và caching phân tán.
* Tuân thủ các tiêu chuẩn bảo mật quốc tế, đảm bảo an toàn cho dữ liệu khách hàng.

**5. ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN**

**5.1. Đánh giá hệ thống hiện tại**

**5.1.1. Tiêu chí đánh giá (theo ISO/IEC 25010)**

Hệ thống ShopGrids được đánh giá dựa trên các đặc tính chất lượng phần mềm:

1. Tính chức năng (Functional Suitability): Đáp ứng đầy đủ yêu cầu nghiệp vụ của TMĐT (quản lý sản phẩm, đơn hàng, thanh toán, gợi ý sản phẩm).
2. Hiệu suất (Performance Efficiency): Thời gian phản hồi nhanh, có khả năng mở rộng xử lý tải cao.
3. Tương thích (Compatibility): Hoạt động trên nhiều nền tảng (web, mobile) và tương thích với các dịch vụ ngoài (API thanh toán, vận chuyển).
4. Khả năng sử dụng (Usability): Giao diện thân thiện, hỗ trợ đa thiết bị.
5. Độ tin cậy (Reliability): Hệ thống có cơ chế dự phòng, nhân bản dữ liệu.
6. Bảo mật (Security): Tuân thủ tiêu chuẩn PCI DSS, ISO/IEC 27001.
7. Khả năng bảo trì (Maintainability): Kiến trúc microservices giúp dễ nâng cấp từng phần.
8. Tính di động (Portability): Có thể triển khai trên nhiều nền tảng cloud khác nhau.

**5.1.2. Phân tích SWOT**

|  |  |
| --- | --- |
| **Yếu tố** | **Mô tả** |
| **S – Strengths** (Điểm mạnh) | - Kiến trúc microservices linh hoạt - Khả năng mở rộng dễ dàng - Triển khai cloud-native, dễ tích hợp dịch vụ ngoài |
| **W – Weaknesses** (Điểm yếu) | - Yêu cầu đội ngũ kỹ thuật trình độ cao - Chi phí vận hành ban đầu cao hơn monolithic - Phức tạp trong quản lý dịch vụ |
| **O – Opportunities** (Cơ hội) | - Nhu cầu TMĐT tăng mạnh toàn cầu - Công nghệ cloud và AI ngày càng phổ biến - Khả năng mở rộng sang thị trường quốc tế |
| **T – Threats** (Thách thức) | - Nguy cơ tấn công mạng tinh vi - Cạnh tranh gay gắt từ các nền tảng TMĐT lớn - Thay đổi chính sách pháp lý và bảo mật dữ liệu |

**5.2. Định hướng phát triển**

**5.2.1. Nâng cấp kiến trúc**

Hoàn thiện kiến trúc microservices với các dịch vụ chuyên biệt (User, Product, Order, Payment, Recommendation, Analytics).

Áp dụng Event-driven Architecture để xử lý luồng công việc bất đồng bộ, giảm tải hệ thống.

**5.2.2. Triển khai đa đám mây (Multi-cloud Deployment)**

Sử dụng kết hợp AWS, GCP và Azure để đảm bảo tính sẵn sàng và tối ưu chi phí.

Thiết lập Disaster Recovery Plan để khôi phục nhanh khi xảy ra sự cố.

**5.2.3. Tích hợp trí tuệ nhân tạo**

Recommendation System: Gợi ý sản phẩm dựa trên Collaborative Filtering và Deep Learning.

Fraud Detection: Phát hiện gian lận giao dịch bằng mô hình học máy giám sát (Supervised Learning).

Predictive Analytics: Dự đoán nhu cầu tồn kho, xu hướng thị trường.

**5.2.4. Tối ưu hiệu năng**

Kết hợp Edge Computing với CDN để xử lý gần người dùng, giảm độ trễ.

Tích hợp GraphQL Federation để tối ưu truy vấn dữ liệu cho ứng dụng web và mobile.

**5.2.5. Bảo mật nâng cao**

Tích hợp Zero Trust Architecture để bảo vệ toàn diện.

Áp dụng Threat Intelligence để phát hiện sớm mối đe dọa.

**5.3. Roadmap triển khai**

|  |  |  |
| --- | --- | --- |
| **Giai đoạn** | **Thời gian** | **Nội dung chính** |
| **Giai đoạn 1** | 0–6 tháng | Hoàn thiện microservices cốt lõi (User, Product, Order, Payment), triển khai trên cloud. |
| **Giai đoạn 2** | 6–12 tháng | Tích hợp Recommendation, Event-driven Architecture, autoscaling Kubernetes. |
| **Giai đoạn 3** | 12–18 tháng | Mở rộng sang multi-cloud, tích hợp AI cho phân tích dữ liệu lớn và dự báo. |
| **Giai đoạn 4** | 18–24 tháng | Hoàn thiện bảo mật Zero Trust, triển khai hệ thống giám sát nâng cao và tối ưu hiệu năng toàn cầu. |

**6. KẾT LUẬN**

**6.1. Tóm tắt nội dung nghiên cứu**

Luận văn đã tiến hành phân tích và thiết kế ShopGrids – một nền tảng thương mại điện tử chuyên biệt cho thiết bị điện tử – dưới góc nhìn hệ thống phân tán (Distributed System). Trên cơ sở lý thuyết về kiến trúc phân tán, microservices và cloud-native, nghiên cứu đã:

1. Xác định các yêu cầu nghiệp vụ và kỹ thuật đối với một nền tảng TMĐT hiện đại.
2. Đề xuất kiến trúc phân tán cho ShopGrids với các thành phần microservices độc lập, cơ sở dữ liệu riêng, giao tiếp qua API Gateway và message broker.
3. Đưa ra giải pháp bảo mật dựa trên tiêu chuẩn ISO/IEC 27001, PCI DSS và khuyến nghị OWASP Top 10.
4. Xây dựng các kỹ thuật tối ưu hiệu năng như caching phân tán, autoscaling Kubernetes, và CDN.
5. Đánh giá hệ thống dựa trên tiêu chí ISO/IEC 25010 và phân tích SWOT, từ đó đề xuất roadmap triển khai theo từng giai đoạn.

**6.2. Đóng góp khoa học và thực tiễn**

Về khoa học:

Cung cấp mô hình tham chiếu cho việc ứng dụng kiến trúc microservices và cloud-native vào lĩnh vực TMĐT, có khả năng đáp ứng khối lượng truy cập lớn và yêu cầu mở rộng linh hoạt.

Đề xuất cơ chế tích hợp AI/ML trong hệ thống phân tán TMĐT để nâng cao trải nghiệm người dùng và hiệu quả vận hành.

Bổ sung nghiên cứu về tối ưu bảo mật và hiệu năng trong môi trường phân tán, kết hợp lý thuyết với triển khai thực tế.

Về thực tiễn:

Mô hình ShopGrids có thể triển khai cho doanh nghiệp vừa và nhỏ với chi phí hợp lý nhưng vẫn đáp ứng tiêu chuẩn quốc tế về an toàn, hiệu năng và khả năng mở rộng.

Giải pháp dễ dàng tích hợp các dịch vụ ngoài như thanh toán, vận chuyển, phân tích dữ liệu.

Khả năng triển khai đa đám mây giúp tăng tính sẵn sàng và độ tin cậy toàn cầu.

**6.3. Hạn chế của nghiên cứu**

Mô hình được thiết kế dựa trên môi trường giả định, chưa được thử nghiệm trên quy mô hàng triệu người dùng thực tế.

Chưa triển khai đầy đủ các tính năng AI nâng cao (Deep Learning-based Recommendation, Fraud Detection).

Chưa phân tích sâu chi phí vận hành khi triển khai đa đám mây.

**6.4. Hướng nghiên cứu và phát triển tiếp theo**

1. Thử nghiệm thực tế trên môi trường cloud công cộng (AWS, GCP, Azure) với dữ liệu và lưu lượng truy cập thực để đánh giá hiệu năng.
2. Tích hợp toàn diện AI/ML vào các quy trình kinh doanh: gợi ý sản phẩm, dự đoán tồn kho, phát hiện gian lận.
3. Tối ưu chi phí vận hành bằng cách áp dụng kiến trúc serverless và các giải pháp tối ưu tài nguyên container.
4. Mở rộng sang đa lĩnh vực ngoài thiết bị điện tử, áp dụng cho các ngành hàng TMĐT khác.
5. Nghiên cứu khả năng áp dụng Blockchain cho quản lý đơn hàng, xác thực nguồn gốc sản phẩm và tăng tính minh bạch giao dịch.