Báo Cáo Kỹ Thuật: Ứng Dụng Web Mua Sắm Phân Tán

1. Giới thiệu

1.1 Tổng quan dự án

- Mục tiêu xây dựng hệ thống web shopping phân tán.
- Yêu cầu chức năng: tìm kiếm sản phẩm, xem chi tiết

1.2 Công nghệ sử dụng

- **Backend & Frontend:** Express.js
- **Cơ sở dữ liệu:** MongoDB (cho dữ liệu sản phẩm)
- **Công cụ tìm kiếm:** Sonic Search Engine (tích hợp qua gRPC)
- **Giao tiếp:** gRPC (giữa main-app và sonic-server), HTTP (cho client)
- **Đóng gói & Triển khai:** Docker, Kubernetes
- **Reverse Proxy & Load Balancer:** Nginx (thông qua Kubernetes Ingress Nginx Controller)
- **Hạ tầng:** Google Cloud Engine (GCE) - được quản lý bởi Terraform

2. Kiến trúc hệ thống

2.1 Tổng quan kiến trúc

- Mô hình microservices với hai thành phần chính: main-app và sonic-server.
- Kiến trúc có thể xem như 3-tier:
    - **Presentation Tier:** main-app phục vụ giao diện người dùng HTML và API.
    - **Application/Logic Tier:** main-app xử lý logic nghiệp vụ, sonic-server xử lý logic tìm kiếm.
    - **Data Tier:** MongoDB lưu trữ dữ liệu sản phẩm, Sonic Engine lưu trữ chỉ mục tìm kiếm.

2.2 Main App (main-app)

- **Chức năng:**
    - Cung cấp giao diện web tĩnh cho người dùng (main-app/public/index.html).
    - Tiếp nhận yêu cầu HTTP từ client (ví dụ: /search).
    - Đóng vai trò gRPC client để giao tiếp với sonic-server cho các tác vụ tìm kiếm.
- **Công nghệ:** Express.js, gRPC, Pino (cho logging).
- **Triển khai:** Dưới dạng Kubernetes Deployment (main-app-deployment) với Service (main-app)

2.3 Sonic Server (sonic-server)

- **Chức năng:**

    - Cung cấp gRPC service (sonic_service.proto) cho main-app để thực hiện:
        - SearchAndFetchItems: Tìm kiếm sản phẩm dựa trên query string và trả về thông tin chi tiết từ MongoDB.
        - IngestData: Đọc dữ liệu từ MongoDB và đẩy vào Sonic Engine để tạo/cập nhật chỉ mục tìm kiếm.
        - GetNodeStatus: Cung cấp thông tin trạng thái của node.
    - Tích hợp và quản lý Sonic Search Engine (child process).
    - Kết nối với MongoDB để truy xuất và đồng bộ dữ liệu sản phẩm.
- **Công nghệ:** Node.js, gRPC, sonic-channel library, Mongoose (cho MongoDB).
- **Triển khai:** Dưới dạng Kubernetes Deployment (sonic-server-deployment) với Service (sonic-server). Mỗi pod sonic-server sẽ chạy cả Node.js application và Sonic engine.

2.4 Hạ tầng Kubernetes

- **Orchestration:** Quản lý life cycle và networking của các container main-app và sonic-server.
- **Networking:**
    - Service (ClusterIP) cho giao tiếp nội bộ giữa main-app và sonic-server.
    - Ingress (ingress-nginx) để expose main-app ra bên ngoài thông qua domain shop.nielnart.io.vn.
- **Configuration:**
    - ConfigMap (sonic-server-config) để quản lý cấu hình của Sonic Engine (sonic-server/sonic.cfg).
    - Secrets (ví dụ: sonic-server-secret cho MONGO_URI và SONIC_AUTH).

3. Đáp ứng các yêu cầu bắt buộc

3.1 Khả năng chịu lỗi (Fault Tolerance)

- **Kubernetes Pod Management:**
    - main-app-deployment và sonic-server-deployment được cấu hình với replicas: 2 (cho sonic-server) và replicas: 2 (cho main-app), đảm bảo có nhiều instance chạy đồng thời.
    - Kubernetes tự động khởi động lại các pod bị lỗi.
- **gRPC Connection (trong main-app):**
    - Logic connectToSonicWithRetry trong main-app/index.js thực hiện kết nối lại với sonic-server theo cơ chế exponential backoff nếu kết nối ban đầu thất bại hoặc bị gián đoạn.
- **Sonic Server:**

- sonic-server khởi chạy Sonic engine như một child process và có logic để theo dõi. Nếu Sonic engine gặp sự cố, sonic-server có thể ghi log và báo cáo status

3.2 Giao tiếp phân tán

- **gRPC (Giao tiếp nội bộ):**
  - main-app giao tiếp với sonic-server qua gRPC (port 50051).
  - Định nghĩa service và message được thực hiện qua file .proto (main-app/proto/sonic_service.proto và sonic-server/proto/sonic_service.proto).
  - Kubernetes Service (sonic-server) đóng vai trò service discovery cho gRPC endpoint.
- **HTTP (Giao tiếp bên ngoài):**
  - Client (trình duyệt) giao tiếp với main-app qua HTTP (port 80 của service, target port 3030 của container).
  - kubernetes/ingress.yaml định nghĩa cách Ingress Nginx Controller expose main-app service ra internet tại shop.nielnart.io.vn.

3.3 Phân mảnh hoặc Sao chép dữ liệu (Sharding or Replication)

- **Application-Level Replication (cho main-app và sonic-server):**
  - sonic-server đều được triển khai với nhiều replicas trên Kubernetes, mỗi pod sẽ có một instance của Sonic engine và lưu trữ data trên trong pod đó(dữ liệu này sẽ tồn tại nếu pod không bị xóa).
- **Data Replication/Consistency (MongoDB):**
  - (Chưa triển khai)
- **Search Index Replication (Sonic):**
  - Mỗi pod sonic-server chạy một instance Sonic engine riêng. Dữ liệu được ingest vào Sonic từ MongoDB.
  - Sonic không hỗ trợ sharding dữ liệu tự động qua nhiều node hay replicas.

3.4 Giám sát / Ghi log đơn giản (Simple Monitoring / Logging)

- **Application Logging:**
  - Cả main-app và sonic-server sử dụng pino logger để ghi log có cấu trúc (xem main-app/index.js và sonic-server/index.js).
  - Log level được cấu hình và có thể phân biệt giữa môi trường development và production.
  - Log bao gồm thông tin về request, gRPC calls, lỗi, và trạng thái kết nối.
- **Kubernetes Monitoring:**

- kubectl logs <pod-name> để xem log của từng pod.
- kubectl describe pod <pod-name> để xem events và trạng thái của pod.
- Kubernetes Dashboard(có thể thông qua port-forward), Kubernetes (Prometheus, Grafana) - Chưa được tích hợp.

3.5 Kiểm tra Stress Test

- **Implementation:**
  - Script main-app/stress-test.js được sử dụng để thực hiện stress test.
  - Script gửi NUM_REQUESTS (5000) yêu cầu HTTP GET đến endpoint /search của main-app với CONCURRENCY_LIMIT (5000).
  - Các search terms được chọn ngẫu nhiên từ một danh sách định sẵn.
- **Observation & Metrics:**
  - Script theo dõi số lượng request thành công và thất bại.
  - Kết quả được in ra console sau khi test hoàn thành.
  - Mục tiêu là quan sát hành vi của hệ thống (CPU/memory usage của pods, response time) khi có nhiều request đồng thời.

4. Các yêu cầu tùy chọn

4.1 Cân bằng tải (Load Balancing)

- **External Load Balancing (Nginx Ingress):**
  - kubernetes/ingress.yaml sử dụng Nginx Ingress controller.
  - default dùng Round Robin để cân bằng tải các request đến main-app service.
- **Internal Load Balancing (Kubernetes Service):**
  - Kubernetes Service (main-app và sonic-server) tự động cân bằng tải các request nội bộ (ví dụ, từ main-app đến sonic-server) giữa các pod backend theo thuật toán round-robin (mặc định).

4.2 Tự động hóa triển khai (Deployment Automation)

- **Containerization (Docker):**
  - Cả main-app và sonic-server đều có Dockerfile (main-app/Dockerfile, sonic-server/Dockerfile) để đóng gói ứng dụng thành Docker image.
  - Các image được đẩy lên Docker Hub (ví dụ: nieltran/main-app:v1.0).
- **Orchestration (Kubernetes):**

    - Toàn bộ hệ thống được định nghĩa bằng các file YAML trong thư mục kubernetes/ (Deployments, Services, Ingress, ConfigMap, HPA, PDB).
    - kubectl apply -f <directory> có thể được sử dụng để triển khai hoặc cập nhật toàn bộ hệ thống.
    - deploy.sh có thể là script để tự động hóa quá trình này.
- **Infrastructure as Code (Terraform):**
    - Thư mục terraform/ cho thấy Terraform được sử dụng để quản lý hạ tầng (ví dụ: máy ảo GCE, network). Điều này giúp tự động hóa việc tạo và quản lý môi trường chạy Kubernetes.

5. Giải thích chi tiết về cách hệ thống hoạt động

5.1 Luồng yêu cầu người dùng (User Request Flow - Tìm kiếm)

1. Người dùng nhập từ khóa tìm kiếm vào giao diện web trên trình duyệt.
2. Trình duyệt gửi HTTP GET request đến http://shop.nielnart.io.vn/search?q=<từ-khóa>.
3. Ingress Nginx controller nhận request, dựa vào host và path, route request đến main-app service.
4. main-app service cân bằng tải request đến một trong các main-app pod.
5. main-app pod (Express.js) nhận request, trích xuất query string.
6. main-app pod (gRPC client) gửi gRPC request SearchAndFetchItems đến sonic-server service.
7. sonic-server service cân bằng tải request đến một trong các sonic-server pod.
8. sonic-server pod (gRPC server) nhận request: a. Query Sonic engine (chạy trong cùng pod) để lấy danh sách ID sản phẩm khớp với từ khóa. b. Dùng các ID này để query MongoDB lấy thông tin chi tiết sản phẩm. c. Trả về danh sách sản phẩm chi tiết cho main-app qua gRPC response.
9. main-app pod nhận gRPC response, xử lý và trả về JSON response cho client qua HTTP.
10. Trình duyệt nhận JSON response và hiển thị kết quả tìm kiếm.

6.1 Điểm mạnh của hệ thống

- **Khả năng chịu lỗi (Resilience):** Kubernetes quản lý pod lifecycle, đảm bảo các service luôn chạy. Logic retry trong main-app tăng cường khả năng phục hồi.
- **Tự động hóa triển khai (Deployment Automation):** Docker, Kubernetes YAML, và Terraform giúp quá trình triển khai và quản lý hạ tầng được tự động hóa.
- **Tìm kiếm hiệu quả (Efficient Search):** Sonic cung cấp khả năng tìm kiếm nhanh và nhẹ.

6.2 Điểm yếu của hệ thống

- **Chưa tích hợp giám sát chuyên sâu (Monitoring & Alerting):**
  Hiện tại hệ thống mới sử dụng logging thủ công bằng kubectl logs. Các công cụ chuyên nghiệp như Prometheus, Grafana hoặc Loki chưa được triển khai, dẫn đến khó giám sát tài nguyên, theo dõi lỗi, và cảnh báo kịp thời.
- **Khả năng mở rộng dữ liệu tìm kiếm còn hạn chế:**
  Sonic Search Engine hiện không hỗ trợ tự động **sharding** hoặc **replication** qua nhiều node. Điều này khiến việc mở rộng hệ thống tìm kiếm trở nên khó khăn khi dữ liệu sản phẩm tăng mạnh.
- **MongoDB chưa có replication hoặc cluster:**
  Cơ sở dữ liệu MongoDB đang hoạt động ở chế độ đơn lẻ (standalone), chưa có cơ chế sao lưu hoặc phân mảnh, làm giảm tính sẵn sàng và khả năng mở rộng.
- **Tính năng người dùng hạn chế:**
  Hệ thống hiện mới chỉ hỗ trợ tính năng tìm kiếm và xem chi tiết sản phẩm. Các tính năng quan trọng khác của một hệ thống mua sắm như giỏ hàng, thanh toán, xác thực người dùng… chưa được tích hợp.
- **Chưa có CI/CD pipeline:**
  Việc triển khai hiện tại vẫn cần thao tác thủ công hoặc thông qua script deploy.sh. Chưa tích hợp CI/CD pipelines như GitHub Actions, GitLab CI, hoặc ArgoCD để tự động hóa toàn bộ quy trình từ build → test → deploy.

7. Kế hoạch phát triển trong tương lai

- **Tích hợp tính năng người dùng nâng cao:**
  - Thêm các module: Đăng nhập/Đăng ký, giỏ hàng, thanh toán, quản lý đơn hàng.
  - Hệ thống quản trị (admin dashboard) để thêm/sửa sản phẩm, theo dõi đơn hàng.
- **Nâng cấp hệ thống tìm kiếm:**
  - Nghiên cứu giải pháp phân mảnh Sonic Engine hoặc thay thế bằng ElasticSearch để hỗ trợ mở rộng theo chiều ngang (horizontal scaling).
  - Cải thiện tốc độ đồng bộ dữ liệu từ MongoDB vào engine tìm kiếm.
- **Triển khai MongoDB Replica Set hoặc sharding:**
  - Tăng độ tin cậy, khả năng phục hồi và hiệu suất truy vấn.
- **Tích hợp hệ thống giám sát & cảnh báo:**
  - Thiết lập Prometheus, Grafana để giám sát CPU, memory, gRPC latency, số lượng request, v.v.
  - Tích hợp Alertmanager để nhận cảnh báo qua email, Slack, Telegram.
- **Áp dụng CI/CD:**

    - Sử dụng GitHub Actions hoặc GitLab CI để build và deploy ứng dụng tự động.
    - Triển khai ArgoCD hoặc FluxCD để hỗ trợ GitOps trên Kubernetes.
- **Bảo mật hệ thống:**
    - Thêm HTTPS (TLS) cho giao tiếp HTTP thông qua Ingress.
    - Giới hạn quyền truy cập tài nguyên trong Kubernetes (RBAC).
    - Bảo vệ gRPC endpoints bằng mã hóa hoặc xác thực token.

8. Kết luận

Hệ thống Web Mua Sắm Phân Tán đã được triển khai thành công trên nền tảng microservices, đáp ứng các yêu cầu cốt lõi về:

- Khả năng chịu lỗi (replica, retry logic),
- Giao tiếp phân tán (HTTP, gRPC),
- Tự động hóa triển khai (Docker, Kubernetes, Terraform),
- Khả năng tìm kiếm nhanh và hiệu quả (Sonic Engine).

Dù còn một số điểm yếu về khả năng mở rộng, tính năng người dùng và giám sát hệ thống, đây là nền tảng tốt để phát triển thêm thành một hệ thống thương mại điện tử đầy đủ tính năng, linh hoạt và có thể triển khai ở quy mô lớn.