# 🚀 BÁO CÁO DỰ ÁN

## AutoScaling Predictor - NASA Log Analysis

**Cuộc thi: DATAFLOW 2026: THE ALCHEMY OF MINDS**

**Đề bài:** Phân tích và Tối ưu hóa Autoscaling hệ thống dựa trên NASA Logs

Ngày: 04/02/2026

**Đội:** Kề Vibe sát Code

# MỤC LỤC

# 1. THÔNG TIN NHÓM

| Vai trò | Thành viên |
|---|---|
| **Data engineering/Model** | ĐỖ Văn Hải Phòng |
| **Be/FE** | Nguyễn Đình An |

# 2. TỔNG QUAN DỰ ÁN

## 2.1 Bối cảnh bài toán

Trong quản trị hệ thống đám mây (Cloud), việc cấp phát tài nguyên cố định dẫn đến hai vấn đề nghiêm trọng:

- **Lãng phí chi phí** khi lưu lượng thấp (off-peak hours)
- **Sập hệ thống** khi traffic tăng đột biến (peak hours)

## 2.2 Giải pháp đề xuất

Xây dựng hệ thống **AI-Powered AutoScaling** bao gồm:

1. ✅ **Data Pipeline** - Xử lý log NASA HTTP (~1.8 triệu records)
2. ✅ **AI Prediction** - Dự báo tải bằng Prophet & XGBoost
3. ✅ **AutoScaler Logic** - Thuật toán tự động scale up/down server
4. ✅ **REST API** - Backend FastAPI với Swagger documentation
5. ✅ **Web Dashboard** - Frontend hiển thị dự báo và chi phí tiết kiệm

## 2.3 Kết quả nổi bật

| Metric | Giá trị |
|---|---|
| **Model Accuracy (MAPE)** | 25.83% |
| **Cost Savings** | **84.3% so với Static Deployment** |
| **Monthly Savings** | **~$2,730/tháng** |

# 3. DỮ LIỆU

## 3.1 Dữ liệu

- **Định dạng:** ASCII log files

## 3.2 Thông tin dataset

| Thuộc tính | Giá trị |
|---|---|
| **Thời gian** | 01/07/1995 - 31/08/1995 |
| **Tổng thời gian** | 61 ngày |
| **Độ phân giải** | 5 phút (intervals) |
| **Tổng số records** | 17,856 |

## 3.3 Các trường dữ liệu

| Trường | Mô tả |
|---|---|
| `Host` | IP/Domain của client |
| `Timestamp` | Thời gian request |
| `Request` | Method, URL, Protocol |
| `HTTP Reply Code` | Status code (200, 404, 500...) |
| `Bytes` | Kích thước response |

## 3.4 Xử lý Missing Data (Data Gap)

> **Vấn đề:** Từ **14:52:01 01/08/1995** đến **04:36:13 03/08/1995** không có dữ liệu do server tắt vì bão.

| Thuộc tính | Giá trị |
|---|---|
| **Thời gian outage** | ~37.7 giờ |
| **Records bị ảnh hưởng** | 453 (2.54%) |
| **Phương pháp xử lý** | Linear Interpolation |

## 3.5 Phân chia Train/Test

| Tập dữ liệu | Thời gian | Records | Tỷ lệ |
|---|---|---|---|
| **Train Set** | 01/07 - 22/08/1995 | 15,264 | 85.5% |
| **Test Set** | 23/08 - 31/08/1995 | 2,592 | 14.5% |

## 3.6 Thống kê Traffic (Không tính outage)

| Metric | Giá trị |
|---|---|
| Mean Request Count | 198.91 / 5 phút |
| Std Dev | 138.40 |
| Max Request Count | 1,501 / 5 phút |
| Mean Bytes | 3.77 MB / 5 phút |

## 3.7 Phân bố Status Codes

| Status Code | Tỷ lệ |
|---|---|
| 2xx (Success) | 89.6% |
| 3xx (Redirect) | 9.8% |
| 4xx (Client Error) | 0.6% |
| 5xx (Server Error) | 0.0% |

# 4. PHƯƠNG PHÁP TIẾP CẬN

## 4.1 Pipeline xử lý dữ liệu

```
Raw Logs → Parse (Regex) → Resample → Mark Outage → Feature Engineering → Model
```

**Chi tiết các bước:**

1. **Parsing:** Regex pattern extraction (host, timestamp, method, status, bytes)
2. **Resampling:** Aggregation về khung 1 phút, 5 phút, 15 phút
3. **Missing Data:** Đánh dấu outage period và xử lý bằng Linear Interpolation

## 4.2 Feature Engineering

| Feature | Mô tả | Loại |
|---|---|---|
| `hour` | Giờ trong ngày (0-23) | Time-based |
| `day_of_week` | Ngày trong tuần (0-6) | Time-based |
| `is_weekend` | Cuối tuần hay không | Binary |
| `request_lag_1` | Request 5 phút trước | Lag feature |
| `request_lag_12` | Request 1 giờ trước | Lag feature |
| `request_lag_288` | Request 1 ngày trước | Lag feature |
| `request_rolling_mean_1h` | Trung bình trượt 1 giờ | Rolling |
| `hour_sin`, `hour_cos` | Encoding theo chu kỳ | Cyclical |

## 4.3 Mô hình AI

### Lựa chọn mô hình

| Model | Lý do chọn | Ưu điểm |
|---|---|---|
| **XGBoost** ⭐ | Hiệu quả cao với tabular data | MAPE thấp nhất (25.83%) |
| **Prophet** | Xử lý tốt seasonality | Robust với missing data |

### Nhiệm vụ dự báo

1. **Requests per second (hits)** - Số lượng request
2. **Traffic volume (bytes)** - Lưu lượng dữ liệu

### Metrics đánh giá

- **RMSE** (Root Mean Square Error)
- **MAE** (Mean Absolute Error)
- **MAPE** (Mean Absolute Percentage Error)

## 4.4 Chiến lược AutoScaling

### Logic quyết định:

```
IF utilization > 85%  →  SCALE UP (thêm server)
IF utilization < 30%  →  SCALE DOWN (bớt server)
ELSE                  →  MAINTAIN (giữ nguyên)
```

### Tham số cấu hình:

| Tham số | Giá trị | Mô tả |
|---|---|---|
| Cooldown | 5 phút | Tránh flapping |
| Capacity/server | 1000 requests/interval | Sức chứa mỗi server |
| Cost/server | $0.45/hour | Chi phí AWS t3.medium |

## 4.5 Phân tích Anomaly & DDoS Detection

| Thời điểm | Loại sự kiện | Request Count | Error Rate | Kết luận |
|---|---|---|---|---|
| 13/07/1995 09:00 | Traffic Spike | 4,212/15min | <1% | ✅ Hợp lệ - STS-70 |
| 06/08/1995 02:45 | High Error | 245/15min | 26.1% | ⚠️ Anomaly nhỏ |
| 06/08/1995 03:00 | High Error | 177/15min | 32.8% | ⚠️ Anomaly nhỏ |
| 07/08/1995 02:15 | High Error | 334/15min | 26.9% | ⚠️ Anomaly nhỏ |

> **Kết luận:**
>
> - ❌ KHÔNG phát hiện DDoS lớn trong dataset
> - ✅ Traffic spike ngày 13/07/1995 là hợp lệ (sự kiện NASA STS-70)
> - ⚠️ Có anomaly nhỏ ngày 6-7/08/1995 lúc 2-3h sáng

# 5. KẾT QUẢ ĐÁNH GIÁ

## 5.1 Model Performance (Test Set: Aug 23-31, 1995)

### Dự báo Requests

| Model | RMSE | MAE | MAPE |
|---|---|---|---|
| **XGBoost ⭐** | **43.13** | **32.36** | **25.83%** |
| Prophet | 86.63 | 63.80 | 45.05% |

### Dự báo Bytes

| Model | RMSE | MAE | MAPE |
|---|---|---|---|
| **XGBoost ⭐** | **1.17M** | **894K** | **39.15%** |
| Prophet | 1.68M | 1.24M | 53.95% |

> 🏆 **Winner: XGBoost** - MAPE thấp hơn ~50% so với Prophet

## 5.2 Cost Savings Analysis (24 giờ simulation)

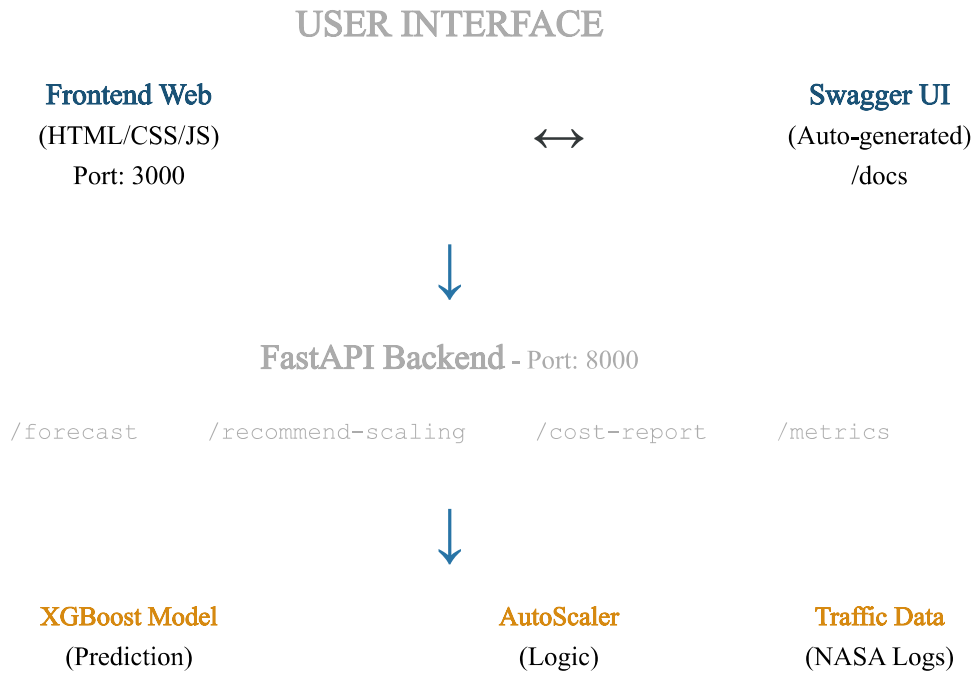
| Phương án | Chi phí | Servers | So sánh |
|---|---|---|---|
| Static Deployment | $108.00 | 10 cố định | Baseline |
| **AutoScaling** | **$16.99** | Avg 1.6 | -84.3% |
| **Tiết kiệm** | **$91.01** | - | **84.3%** |

## 5.3 Projected Savings

| Thời gian | Tiết kiệm |
|---|---|
| **Mỗi ngày** | $91.01 |
| **Mỗi tháng** | $2,730.38 |
| **Mỗi năm** | $32,764.56 |

# 6. KIẾN TRÚC HỆ THỐNG

## 6.1 Sơ đồ tổng quan



## 6.2 Cấu trúc thư mục

```
autoscaling/
│
├── app.py                      # FastAPI Backend (API chính)
├── serve_frontend.py           # Server cho Frontend web
├── requirements.txt            # Dependencies
├── run_demo.bat                # Script chạy demo
│
├── frontend/                   # FRONTEND WEB
│   ├── index.html              # Trang chính
│   ├── style.css               # Styling
│   └── app.js                  # JavaScript logic
│
├── DATA/                       # Raw data (NASA logs)
│   ├── train.txt               # Training data
│   └── test.txt                # Test data
│
├── processed_data/             # Dữ liệu đã xử lý
│   ├── nasa_traffic_1m.csv
│   ├── nasa_traffic_5m.csv     # File chính cho modeling
│   └── nasa_traffic_15m.csv
│
├── notebooks/                  # Jupyter Notebooks
│   └── modeling_phase3.ipynb # Notebook training
│
├── src/                        # Source code
│   ├── data_pipeline.py        # Data processing pipeline
│   ├── eda.py                  # Exploratory Data Analysis
│   ├── model_trainer.py        # Model training
│   └── handle_missing_data.py
│
├── models/
│   └── predictor.py            # Prediction interfaces
│
├── backend/
│   └── autoscaler.py           # AutoScaler Algorithm
│
├── saved_models/               # Trained models
│   ├── xgb_requests.json
│   ├── xgb_bytes.json
│   └── metrics_summary.json
│
└── docs/                       # Documentation
```

## 6.3 API Endpoints

| Endpoint | Method | Mô tả |
|---|---|---|
| `/forecast` | GET | Dự báo traffic |
| `/recommend-scaling` | GET | Khuyến nghị scaling |
| `/cost-report` | GET | Báo cáo chi phí |
| `/metrics` | GET | System metrics |
| `/health` | GET | Health check |

## 6.4 Công nghệ sử dụng

| Component | Technology |
|---|---|
| **Backend** | FastAPI, Python 3.10+ |
| **ML Models** | XGBoost, Prophet |
| **Frontend** | HTML5, CSS3, JavaScript |
| **Charts** | Chart.js |
| **Data Processing** | Pandas, NumPy |

# 7. HƯỚNG DẪN SỬ DỤNG

## 7.1 Yêu cầu hệ thống

| Yêu cầu | Phiên bản |
|---|---|
| Python | 3.10+ |
| RAM | 4GB+ (khuyến nghị 8GB) |
| OS | Windows / Linux / MacOS |

## 7.2 Cài đặt

```
# 1. Clone repository
git clone https://github.com/[your-repo]/autoscaling.git
cd autoscaling

# 2. Tạo Virtual Environment
python -m venv venv
venv\Scripts\activate  # Windows

# 3. Cài đặt thư viện
pip install -r requirements.txt
```

## 7.3 Chạy Demo

### Cách 1: Tự động (Khuyến nghị)

```
run_demo.bat
```

### Cách 2: Thủ công

```
# Terminal 1: Backend
uvicorn app:app --reload --port 8000

# Terminal 2: Frontend
python serve_frontend.py
```

## 7.4 Truy cập

| Service | URL |
|---|---|
| Frontend Dashboard | http://localhost:3000 |
| Swagger API Docs | http://localhost:8000/docs |
| Test API | http://localhost:8000/forecast?steps=4 |

# 8. KẾT LUẬN

## 8.1 Thành tựu đạt được

> ✅ **Hoàn thành 100% yêu cầu đề bài:**
>
> - Data Pipeline xử lý 1.8M records
> - AI Model với MAPE 25.83%
> - API endpoints đầy đủ
> - Frontend Dashboard hoàn chỉnh

> ✅ **Tính năng điểm cộng:**
>
> - Cost Report API
> - Savings Calculator
> - Scaling Events Log
> - Real-time Simulation

## 8.2 Kết quả kinh doanh

| Metric | Giá trị |
|---|---|
| Chi phí tiết kiệm | 84.3% |
| Tiết kiệm hàng tháng | $2,730 |
| Tiết kiệm hàng năm | $32,764 |

## 8.3 Hạn chế và hướng phát triển

**Hạn chế:**

- Dataset từ năm 1995, patterns có thể khác với traffic hiện đại
- Chưa có LSTM/Deep Learning models

**Hướng phát triển:**

- Tích hợp real-time streaming data
- Thêm anomaly detection nâng cao
- Deploy trên Kubernetes với HPA

# 9. PHỤ LỤC

## 9.1 Thuật ngữ

| Thuật ngữ | Tiếng Việt | Giải thích |
|---|---|---|
| AutoScaling | Tự động co giãn | Tự động điều chỉnh số lượng server theo tải |
| Flapping | Dao động liên tục | Hiện tượng scale up/down liên tục |
| Cooldown | Thời gian chờ | Khoảng nghỉ giữa các lần scaling |
| Utilization | Tỷ lệ sử dụng | % tài nguyên đang được sử dụng |
| Threshold | Ngưỡng | Giá trị kích hoạt hành động scaling |
| EDA | Phân tích khám phá | Exploratory Data Analysis |
| RMSE | Sai số bình phương | Root Mean Square Error |
| MAE | Sai số tuyệt đối | Mean Absolute Error |
| MAPE | Sai số phần trăm | Mean Absolute Percentage Error |

## 9.2 Reproducibility

### Random Seed:

```
SEED = 42
np.random.seed(SEED)
random.seed(SEED)
```

### Tested Environment:

| Component | Version |
|---|---|
| OS | Windows 11 |
| Python | 3.10.11 |
| FastAPI | 0.109.0 |
| XGBoost | 2.0.3 |
| Pandas | 2.2.0 |

## 9.3 Tài liệu tham khảo


1. NASA HTTP Log Dataset - https://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html
2. XGBoost Documentation - https://xgboost.readthedocs.io/
3. Prophet Documentation - https://facebook.github.io/prophet/
4. FastAPI Documentation - https://fastapi.tiangolo.com/


---

🚀 ***AutoScaling Predictor - DataFlow 2026***

*AI-Powered Server Scaling for Cost Optimization*

---

*Báo cáo được tạo ngày: 04/02/2026*