Root Cause Analysis of Cloud Microservices

Supervised by
Dr. Yan Liu

-Gayathiri Elambooranan

-Pranay Sood

Integration of "The PetShop Dataset" in Root Cause Analysis Project

- **Key Resource from the Paper:**

•The PetShop Dataset: Purpose-built to facilitate root cause analysis in microservice environments. It offers a rich set of metrics (latency, requests, availability) collected from a distributed application simulating real-world microservice performance issues

- **Dataset Utilization**:
    1. Data-Driven Insights: Employed the dataset to the LLMs, leveraging the data to simulate real-world microservice performance issues.

Ground Truth Utilization: Applied ground truth data provided in the dataset for validating model predictions, crucial for assessing the zero-shot learning capabilities of LLMs.

- **Utility and Impact:**

Research Acceleration: By providing a ready-to-use dataset, the paper allows researchers and developers to concentrate on creating and refining analysis techniques, speeding up innovation in cloud incident management.
Benchmarking Tool: The dataset serves as a vital benchmark for comparing the effectiveness of various analytical methods, enhancing the understanding of different RCA approaches in cloud computing environments.

Citation: Hardt, M., Orchard, W., Blöbaum, P., Kasiviswanathan, S., & Kirschbaum, E. (2023). The PetShop Dataset — Finding Causes of Performance Issues across Microservices. Link to paper

Research Goals:

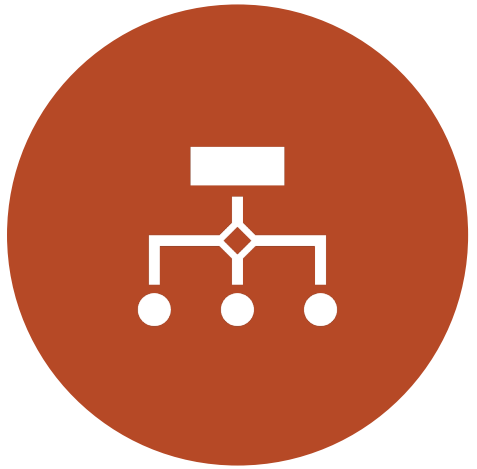
Optimize Data for LLMs: Refine and structure datasets for optimal use with large language models, focusing on enhancing data compatibility for advanced analytics.

Implement Zero-Shot Learning: Explore and compare the effectiveness of zero-shot learning using LLMs and traditional models in analyzing cloud incidents without prior training.

Evaluate Model Performance: Conduct systematic evaluations of the responses generated by LLMs to assess their accuracy and applicability in cloud incident management scenarios.

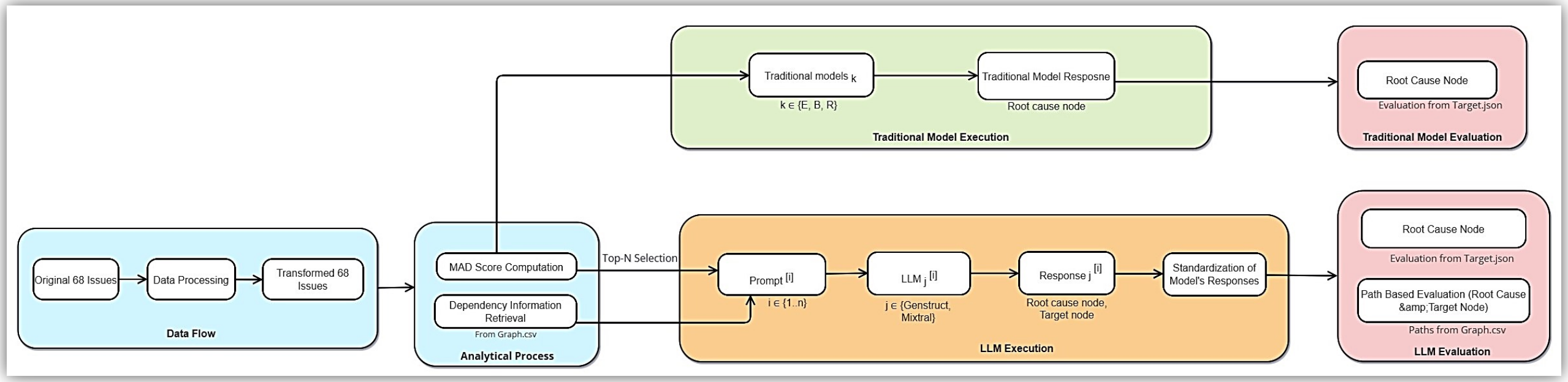
Week’s Research Goals:

IMPLEMENT THE WORKFLOW FOR ALL THE 68 ISSUES IN THE DATASET.



EVALUATE THE RESPONSES GENERATED BY THE MODELS.

Project Workflow:



Project Workflow:

1. Data Flow
Original Issue Identification: The process begins with 68 original issues that are documented for analysis.
Data Processing: These issues are then transformed into a more suitable format for analysis, resulting in the "Transformed 68 Issues" file.

2. Analytical Process
MAD Score Computation: Each transformed issue is evaluated to calculate a Mean Absolute Deviation (MAD) Score, which helps prioritize issues based on their severity.
Dependency Information Retrieval: Essential dependency information is retrieved from Graph.csv, aiding in the detailed analysis of issues.
Top-N Selection: The most critical issues are selected based on their MAD Scores for detailed examination.

3. Model Execution
Traditional Model Execution:
Prompt Generation: Create prompts for the top selected issues.
Model Processing: Analyze prompts using traditional models ($k \in \{E, B, R\}$) to identify root causes.
Model Response: Focus on identifying the root cause for each issue.

LLM Execution:
Prompt Processing: Process prompts through Large Language Models (LLMs) like Genstruct and MixtraL.
Response and Analysis: LLMs provide comprehensive responses, including root and target node identification.
Standardization of Model's Responses: Manually standardize responses to facilitate comparison during evaluation.

4. Evaluation

Traditional Model Evaluation:
Root Cause Analysis: Verify the accuracy of root cause identification against standards in Target.json.
LLM Evaluation:
Comprehensive Evaluation: Assess the accuracy and connectivity of root and target nodes using Graph.csv..

Data Preparation for Reproducibility:

Original Dataset Overview:

microservice	169.254.170.2_remote	AWSSimpleSystem	DynamoDB	Evidently_A	PGSQL Qu	PetAdoptio	PetSearch_	PetSearch_	PetSite	SSM_AWS:	STS_AWS::	Servi-searc	SimpleNoti	SimpleSyst	AWS::Stepl	StepFnStat	StepFuncti	adoptions@	amazon.co	https://sqs
metric	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency	latency
statistic	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average	Average
unix_timestamp																				
1681857120		0.022925586	0.07992	0.298486	0.017294	0.061989	0.00655	0.00501	0.089822	0.207806		0.010186		0.014786	3.189154	3.189154		0.006477	0.069228	
1681857420		0.025099453	0.086155	0.293033	0.016698	0.065394	0.006562	0.004108	0.101388	0.171568		0.010325		0.015578	3.096909	3.096909		0.00751	0.064923	
1681857720	0.000542223	0.022688498	0.064216	0.290658	0.018406	0.056862	0.179905	0.048261	0.291667	0.199908		0.008529		0.015915	2.138909	2.138909		0.005479	0.068525	
1681858020		0.02099214	0.092376	0.273948	0.016851	0.064092	0.30402	0.134467	0.542386	0.203831		0.010938	0.018746	0.014825	2.67	3.227		0.00851	0.061852	0.007441
1681858320		0.021664079	0.059999	0.256975	0.015931	0.056485	0.247262	0.164646	0.398525	0.179644		0.473379	0.018009	0.015132	3.069941			0.008301	0.066562	0.008378

Original Dataset Features:

- Rows: 8
- Columns: 295
- Complex Structure: Contains multiple metrics and microservices across many columns.
- Mixed Data Types: Includes metadata (metric type, statistic) and actual data values in the same DataFrame, complicating analysis.
- Granular Metrics: Each microservice is associated with multiple metric columns, adding complexity.

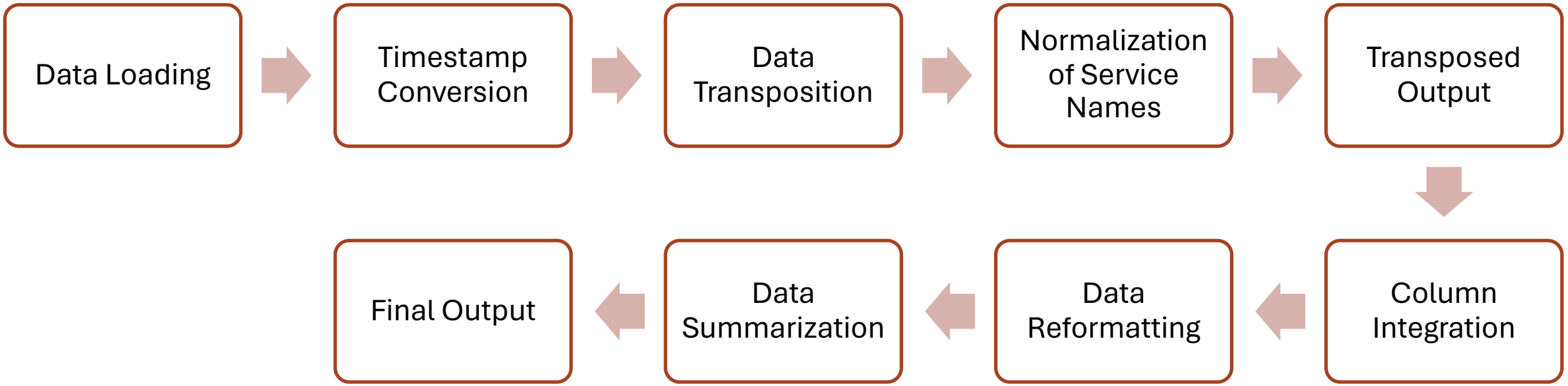
Data Processing for Reproducibility

Transformed Dataset Overview:

microservice	timestamp	availability_Average	latency_Average	latency_p50	latency_p90	latency_p95	latency_p99	requests_Sum
169.254.170.2_remote	2023-04-18 22:32	0	0	0	0	0	0	0
169.254.170.2_remote	2023-04-18 22:37	0	0	0	0	0	0	0
169.254.170.2_remote	2023-04-18 22:42	100	0.000542223	0.000537004	0.000723743	0.000726765	0.000729191	4
169.254.170.2_remote	2023-04-18 22:47	0	0	0	0	0	0	0
169.254.170.2_remote	2023-04-18 22:52	0	0	0	0	0	0	0
AWS::StepFunctions::StateMachine	2023-04-18 22:32	76.92307692	3.189153886	3.397865049	5.582957546	5.604437532	5.621681008	13
AWS::StepFunctions::StateMachine	2023-04-18 22:37	54.54545455	3.096909067	3.88907003	4.471176208	4.52921137	4.5761814	11
AWS::StepFunctions::StateMachine	2023-04-18 22:42	81.81818182	2.138909078	1.429626433	4.266421352	4.321850618	4.366712081	11
AWS::StepFunctions::StateMachine	2023-04-18 22:47	75	2.669999983	2.992374046	4.393585673	4.468159901	4.52872962	8
AWS::StepFunctions::StateMachine	2023-04-18 22:52	52.94117647	3.069941125	3.411385675	4.544162108	6.139953197	6.250144441	17
AWSSimpleSystemsManagement_AW	2023-04-18 22:32	100	0.022925586	0.023173356	0.032195391	0.033399313	0.034389235	32
AWSSimpleSystemsManagement_AW	2023-04-18 22:37	100	0.025099453	0.024304421	0.038403443	0.04580856	0.047954327	34
AWSSimpleSystemsManagement_AW	2023-04-18 22:42	100	0.022688498	0.020086298	0.033287497	0.035246561	0.037403392	24
AWSSimpleSystemsManagement_AW	2023-04-18 22:47	100	0.02099214	0.019707039	0.03282339	0.033792758	0.034588825	24
AWSSimpleSystemsManagement_AW	2023-04-18 22:52	100	0.021664079	0.021268434	0.029408349	0.034746243	0.035751237	30
DynamoDB_AWS::DynamoDB	2023-04-18 22:32	100	0.079919875	0.060105718	0.083109547	0.31914865	0.31977116	12
DynamoDB_AWS::DynamoDB	2023-04-18 22:37	100	0.086155306	0.060105718	0.215564762	0.244804823	0.250215628	11
DynamoDB_AWS::DynamoDB	2023-04-18 22:42	100	0.064216267	0.062292828	0.077856085	0.078081293	0.078261928	11
DynamoDB_AWS::DynamoDB	2023-04-18 22:47	100	0.092376113	0.079241844	0.218137176	0.219156819	0.219975963	8
DynamoDB_AWS::DynamoDB	2023-04-18 22:52	100	0.059999143	0.060971193	0.077661564	0.078179327	0.078596022	17

- **Transformed Dataset Features:**
- Rows: 210
- Columns: 9
- Simplified Structure: Each row aligns a microservice with a timestamp, greatly simplifying the data structure.
- Consolidated Metrics: Reduces multiple metric columns into fewer, comprehensive columns (e.g., latency_Average).
- Long Format: Adapts data into a long format ideal for detailed statistical analysis and machine learning applications.

Data Processing Flow:



Data Processing Flow:

The systematic steps used to process CSV files for data analysis using Python and Pandas, focusing on the transformation and summarization of the data.

- All 68 Issues from all four scenarios were transformed using the script.

Transposing CSV Files:

- Data Loading: Loads CSV data into a DataFrame to prepare for processing.
- Timestamp Conversion: Converts Unix timestamps to a readable date format.
- Data Transposition: Adjusts the DataFrame structure by switching rows and columns, making each microservice a column header.
- Normalization of Service Names: Cleans up service names by removing trailing numbers and special characters.
- Output Creation: Saves the adjusted and transposed data for further manipulation.

Transforming Data:

- File Loading for Transformation: Confirms data availability in transposed files and proceeds with transformations.
- Column Integration: Merges specific columns to create a new composite column that enhances data description.
- Data Reformatting: Changes the dataset into a long format, making it easier for analysis and manipulation.
- Data Summarization: Employs pivot tables to aggregate and summarize data values based on microservice and timestamp.
- Final Output: Outputs the comprehensively transformed data, ready for analysis or reporting.

Benefits of Transformation:

Enhanced Data Usability: Streamlines data, making it easier to manage and analyze.
Standardized Processing: Ensures uniform data processing, facilitating reproducible and consistent analysis across multiple runs.
Reduced Preprocessing for ML: Clean and structured data minimizes the need for preprocessing, reducing potential errors and variances.

LLM Methodology: