Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010401
ISSN 2790-0916 Vol. 1, Issue 4: 1-9

# *Distributed System for Wheel and Rail Detection Based on ZeroMQ*

**Kavita Sahhil***

*Myanmar Institute of Information Technology, Myanmar*

**corresponding author*

***Keywords:*** ZeroMQ, Wheel/Rail Detection, Distributed System, Wheel/Rail Force

***Abstract:*** With the continuous development of network technology, distributed systems have been widely studied and applied. The Publish/Subscribe (Pub/Sub) model supports asynchronous and multipoint communication, enabling the participants of distributed systems to be completely decoupled in time, space, and control flow, thereby solving the problems of integration and dynamic reorganization of distributed systems. The purpose of this paper is to design a distributed system for wheel and rail detection based on ZeroMQ. On the basis of distributed system research, it first analyzes the ZEROMQ message queue operation in detail, combining the characteristics of big data, system architecture, logical architecture, partial architecture and ZERCMQ-based. The distributed design scheme of the distributed system is proposed, and a set of methods to solve the lateral force and vertical force of the wheel-rail contact based on the axonometric method and the combined wheel measurement method are proposed. Finally, the final performance test of the distributed system based on ZEROMQ, the insert performance results show that when the number of insert commands is 10, the average execution time is 2,500,000us, and the query performance results show that when the number of search commands is 10, the average The execution time is 500000us, it can be seen that the system has excellent concurrent performance.

## 1. Introduction

With the rapid development of cloud storage and cloud computing technology, data storage, indexing and search have become the main technical bottlenecks in Internet data centers. This is because the current Internet data volume is large, and the relational database based on the ACID scheme is difficult to meet the current data management needs. Therefore, distributed storage has become the most important way to store and manage data in the world [1]. Physically distributed systems can scale quickly and horizontally, but the interactions between systems are becoming more and more complex. And the transmission speed is very fast, so the best message system is a key factor to understand the distribution system [2].

There are many researches on distributed systems at home and abroad. Ofenloch A provides CPS with an event-based distributed simulation environment that is reusable and easily scalable across various organizations. Simulations are performed using a software model that simulates the functional behavior of the CPS to be tested. Regardless of the platform or programming language chosen, simulation models or interface adapters for hardware components can be developed using defined software interfaces. They can be integrated into a simulated environment and executed on a distributed computer system with minimal effort, while communicating via ZeroMQ. This simulation environment is particularly suitable for systems that require low latency to guarantee real-time performance [3]. Conradt T compared three different strategies for deriving sub-watershed evaporation: (1) simulated by SWIM without any spatial calibration, (2) derived from remotely sensed surface temperature, and (3) calculated from long-term precipitation and emissions data. The results show some agreement between modeled and remote sensing-based evapotranspiration rates, but there appears to be no correlation between remote sensing and water balance-based estimates. Subsequent analyses of individual subwatersheds identified systematic error amplification in input weather data and in inter-instrument flow calculations as sources of uncertainty. The results encourage careful utilization of disparate data sources to enhance distributed hydrological modeling [4]. Kargarian A reviews distributed/decentralized algorithms for solving the optimal power flow (OPF) problem in power systems. Six decomposition coordination algorithms are studied, including Analytical Objective Cascade (ATC), Optimality Conditional Decomposition (OCD), Alternating Direction of Multipliers (ADMM), Auxiliary Problem Principle (APP), Consensus+Innovation (C+I) and Proximity Messaging (PMP). Basic concepts, general formulas, applications of DC-OPF, and solutions for each algorithm are introduced. We apply these six decomposition coordination algorithms to a test system and discuss their key properties and simulation results [5]. Based on the relevant research at home and abroad, although there are many methods to describe the system modeling, there is a lack of corresponding research on the description of the distributed system based on ZeroMQ.

This paper aims to provide effective help for the research of wheel-rail detection system by developing a distributed system of wheel-rail detection based on ZeroMQ. Distributed system simulation can not only flexibly provide simulation data under various external environments (such as: clutter environment, interference environment and complex terrain environment) for current research, but also meet the requirements of advanced signal processing technology and data processing technology in advance research verification requirements. The construction of distributed modeling and simulation platform will greatly improve the authenticity and scalability of system simulation, and it is also of great significance for training operators and assisting software designers to complete software function tests.

## 2. Research on Distributed System of Wheel and Rail Detection Based on ZeroMQ

### 2.1. Overview of ZeroMQ and its Performance

ZeroMQ is an open source message queue project developed by the American iMatix company, which integrates network exceptions, asynchrony, buffers, multithreading, etc. ZeroMQ sends and receives network information in partial messages, saving a lot of code. ZeroMQ is a set of intelligent transport layer protocols that can help developers create free shared programs [6-7]. Its main features are as follows:

Support high concurrency asynchronous Socket protocol, faster than traditional TCP protocol, suitable for distributed simulation and transmission system; no lock between processes, use special background thread to process message data, good parallel performance; components can be started in any order, Components can be added or withdrawn freely; it includes an automatic message queuing mechanism, and queues that are too slow are automatically buffered or discarded; a large number of demos are used to reduce the user's learning cost and difficulty of use; through the combination of various internal modes, the distributed system can be reduced The construction difficulty [8-9].

### 2.2. Distributed System

The so-called "distributed system" can be understood as a computer system based on computers connected to the Internet at the time of design and research, and the system will include many shared computers. Among them are many material or intellectual resources [10-11]. These resource units have a high degree of autonomy while cooperating with each other, which can better manage the resources within the system. On this basis, services or related activities can be strongly differentiated, and even multiple distribution systems can be carried out simultaneously. As a new type of multi-computer system, distributed computer system pays great attention to the overall allocation, management, operation and task allocation of resources [12-13]. The resource part does not refer to a large number of physical devices, but to a large number of intellectual resources. It mainly covers communication interfaces, processors, backup memory and input/output devices. The latter consists of processes, databases, tables, files, and processes. These resources are allocated to each physical node, and the communication between each node is mainly through the network, so that a unified computer system can be created [14-15].

### 2.3. Force Analysis of Wheel-rail Contact Force

The force between the wheel and rail can be decomposed into three vertical components, namely lateral force, vertical force and tangential force. The vertical force acts as a support, the lateral force acts on the lateral force when the wheels of the train rotate, and the longitudinal force acts on the running train. Since the three-way wheelchair will have a professional impact on the actual measurement, and the wheel angle will also have a corresponding impact on the actual measurement result, the patch method requires the use of wheelchair guide rails [16-17]. For investigations of actual power levels, choose the appropriate method to resolve the ruler after determining the measurement location. In wheel dynamometers based on shaft and plate stress measurements, lateral and vertical forces often affect the measured stress simultaneously, so it is necessary to decouple lateral and vertical stress measurements [18].

## 3. Design and Research of Distributed System for Wheel and Rail Detection Based on ZeroMQ

### 3.1. The Overall Structure of the System

The whole prototype system is divided into two parts: the master control end and the node agent end. The main control terminal is mainly used for the modeling of the distributed system based on DDS, the generation of components and the command control in the simulation operation; the node proxy terminal exists on each node, which is used to interact with the main control terminal and receive the information from the main control terminal. Commands control the operation of components on the node and collect information. The prototype system mainly includes the following functions:

Distributed system modeling and maintenance function: Model the DDS-based distributed system through a visual drag-and-drop method, and generate system description documents & schematic diagrams of the entire system as required: The distributed system model is maintained.

Distributed system automatic generation module: It can automatically generate all componentization and corresponding configuration files in the distributed system according to the specification requirements of the resources of the components in the description document.

Simulation operation function: It can start the distributed system according to the tasks defined in the description document and switch tasks through commands, control the operation of the monitoring module on the node and save the monitoring information to the database, which is convenient for later analysis.

Component resource information collection function: In order to evaluate the running status of the entire distributed system, it is necessary to collect the resource occupancy information of the components in real time, and save the collected information to the database for later analysis.

### 3.2. The Principle of Wheelset Measurement Based on Wheel Strain

Attach a strain gage to the object under test, and the strain gage produces a strain corresponding to the strain of the object under test. The corresponding resistance changes, so that the relevant data of the measuring instrument can be determined.

The resistance change rate of a general voltmeter is a constant value, and the corresponding voltage is linear.

$$\frac{\Delta R}{R} = \mu\varepsilon \tag{1}$$

where R is the resistance value of the voltmeter.

For unknown interference that may occur when measuring the rail-rail contact force in the actual patch, the patch side bridge needs to be selected. Generally, the general current group bridge to be tested is selected as the full bridge, and the obtained bridge output is:

$$\rho = 1/4KE(\varepsilon_1 + \varepsilon_2 - \varepsilon_3 - \varepsilon_4) \tag{2}$$

where μ is the constant value ε strain.

### 3.3. System Module Architecture Design

The whole system is divided into three levels, namely the third-party library layer, DB layer and Broker layer.

Insert module: Receive annoying data and distribute the data to each back-end database. The specific distribution strategy will depend on the distributed strategy and the performance of the back-end database. That is, the distributed strategy is based on the performance of the back-end database, such as information such as cpu, memory, disk and load.

HTTP JSON analysis module: This module receives data queries in JSON format, and these requests are passed through the HTTP WEB service module.

Query module: The query module sends all query requests to the back-end, and receives the response from the back-end. When all the back-end reply information is received, the information is merged and sent to the JSON analysis module.

Load balancing: This module receives the information of all the back-end databases transmitted by the monitoring module, performs load balancing algorithm according to the information, and obtains the back-end load balancing sequence, which is convenient for querying and inserting modules.

Monitoring module: This module receives the heartbeat and system information of all back-end servers and other brokers, maintains the status information and system information of all online nodes, and advertises this information to other modules.

## 4. Experiment and Test of Wheel-rail Detection Distributed System Based on ZeroMQ

### 4.1. Experimental Environment

In order to verify the feasibility and functionality of the prototype tethering, the experiment was tested in a 200M LAN environment. The entire local environment includes five machines, whose specific configurations are shown in Table 1. In order to verify factors such as background traffic and the performance of the distributed system, the Wire Shaik packet capture tool will be deployed on the node to collect the network traffic in the distributed system in real time. Use Matlab and SPSS tools to qualitatively and quantitatively analyze the collected data, and generate corresponding charts.

*Table 1. Machine configuration table in the experimental environment*

| CPU name | Host configuration | Operating system | Deploy |
|---|---|---|---|
| Z1 | Pentiun(R) G620 2.6GHz 8GB RAM | Windows 10 | Master control |
| Z2 | Pentiun(R) G620 2.6GHz 8GB RAM | Windows 10 | Proxy |
| Z3 | Pentiun(R) G620 2.6GHz 8GB RAM | Windows 10 | Proxy |
| Z4 | Pentiun(R) G620 2.6GHz 8GB RAM | Windows 10 | Proxy |
| Z5 | Pentiun(R) G620 2.6GHz 8GB RAM | Windows 10 | Proxy |

### 4.2. Analysis of Test Results

As shown in Table 2, the insertion performance of multi-thread concurrency (simultaneous read and write, one word query) and the query performance of multi-thread concurrency (simultaneous read and write, one word query) are shown. As can be seen from the figure, the system has excellent concurrent performance.

*Table 2. Adding performance and query capabilities when concurrent*

| | Add performance | Query performance |
|---|---|---|
| 10 | 2500000 | 500000 |
| 100 | 1000000 | 1000000 |
| 1000 | 2000000 | 2300000 |
| 10000 | 5200000 | 9000000 |
| 100000 | 17500000 | 11000000 |
| 1000000 | 27000000 | 362000000 |

As shown in Table 2, when the number of insert commands is 10, the average execution time is 2,500,000us; when the number of insert commands is 100, the average execution time is 1,000,000us; when the number of insert commands is 1,000, the average execution time is

2,000,000us; the number of insert commands When it is 10000, the average execution time is 5200000us; when the number of insert commands is 100000, the average execution time is 17500000us; when the number of insert commands is 1000000, the average execution time is 27000000us.


*Figure 1. Add performance and query function comparison when concurrent*

As shown in Figure 1, when the number of search commands is 10, the average execution time is 500000us; when the number of search commands is 100, the average execution time is 500000us; when the number of search commands is 1000, the average execution time is 1000000us; the number of search commands When it is 10000, the average execution time is 2300000us; when the number of search commands is 100000, the average execution time is 11000000us; when the number of search commands is 1000000, the average execution time is 27000000us.

As shown in Table 3, the insertion performance and query performance of multi-thread concurrency (simultaneous reading and writing, two-word query) are shown. As can be seen from the figure, the system has excellent concurrent performance.

*Table 3. Insert performance and query performance when multi-threaded concurrently*

| | Add performance | Query performance |
|---|---|---|
| 10 | 24000000 | 375000000 |
| 20 | 13000000 | 10000000 |
| 50 | 14000000 | 25000000 |
| 100 | 15000000 | 100000000 |
| 200 | 26000000 | 110000000 |
| 500 | 26000000 | 130000000 |

As shown in Table 3, when the number of insert commands is 10, the average execution time is 24000000us; when the number of insert commands is 20, the average execution time is 13000000us; when the number of insert commands is 50, the average execution time is 14000000us; the number of insert commands When it is 100, the average execution time is 15000000us; when the number of insert commands is 200, the average execution time is 26000000us; when the number of insert commands is 500, the average execution time is 26000000us.

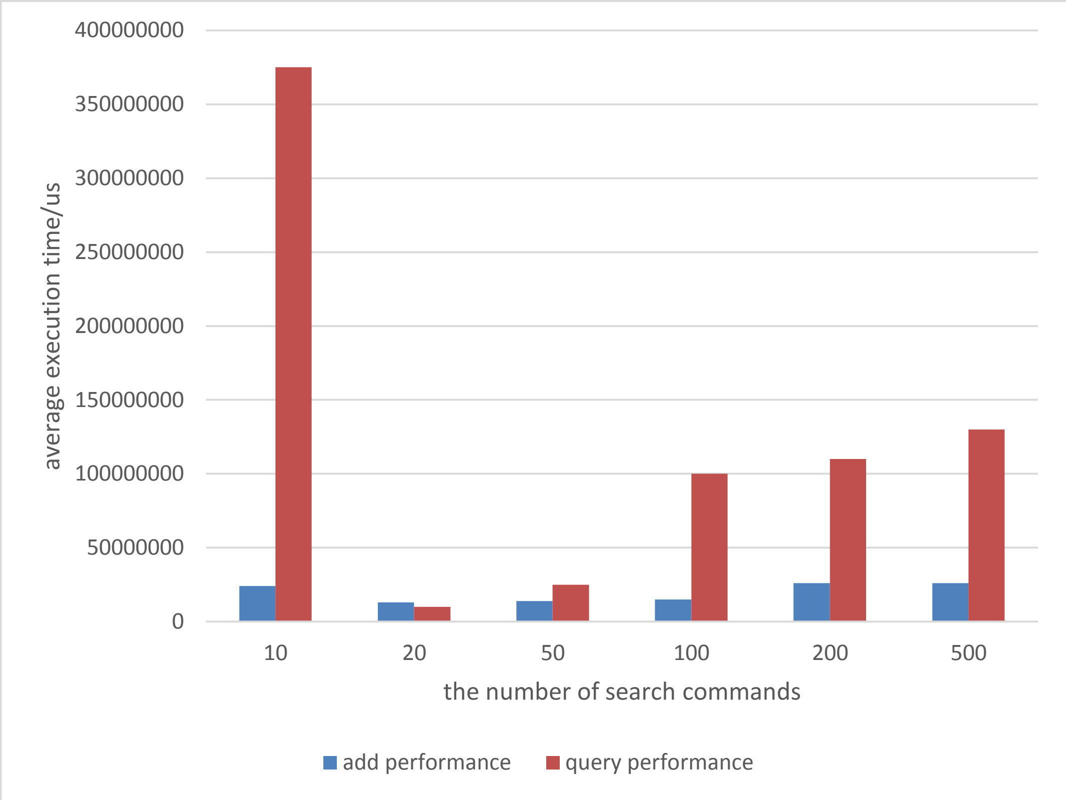


*Figure 2. Comparison of insert performance and query performance with multi-threaded concurrency*

As shown in Figure 2, when the number of search commands is 10, the average execution time is 500000us; when the number of search commands is 20, the average execution time is 375000000us; when the number of search commands is 50, the average execution time is 25000000us; the number of search commands When it is 100, the average execution time is 100000000us; when the number of search commands is 200, the average execution time is 110000000us; when the number of search commands is 500, the average execution time is 130000000us.

## 5. Conclusion

In the key real-time fields such as national defense and aerospace, the performance and reliability of the system have strict requirements, so the design optimization of the system has put forward higher requirements. How to simulate the design scheme of the system and analyze the relative performance of the system in the design stage of the system, and use it as a reference to guide the iterative design and implementation of the system is particularly important. Therefore, the main purpose of this paper is to provide an extensible description method, component resource occupation model and component interaction model for ZeroMQ-based distributed systems. Based on this, it can automatically generate a ZeroMQ-based The components required by the distributed system, through the definition of the task, start the operation of the system for simulation, and solve the modeling and simulation problems of the distributed system based on ZeroMQ in a specific field. In the system design stage, the data collected from the operation of the distributed system composed of the automatically generated simulation components can be used to analyze and calculate the relevant performance parameters to guide the design of the system. After the system is designed and put into operation, the performance of the device can be verified based on the collected data and the performance of the real system can be evaluated.

## Funding

This article is not supported by any foundation.

## Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

## Conflict of Interest

The author states that this article has no conflict of interest.

## References

*[1] Pichlik P, Bauer J. Adhesion Characteristic Slope Estimation for Wheel Slip Control Purpose based on UKF. IEEE Transactions on Vehicular Technology, 2020, PP(99):1-1.*

*[2] Parise F, Gentile B, Lygeros J. A distributed algorithm for average aggregative games with coupling constraints. IEEE Transactions on Control of Network Systems, 2020, 7(2):770-782.*

*[3] Ofenloch A, Greif F. A Flexible Distributed Simulation Environment for Cyber-Physical Systems Using ZeroMQ. Journal of Communications, 2018, 13(6):333-337.*

*[4] Conradt T, Wechsung F, Bronstert A. Three perceptions of the evapotranspiration landscape: comparing spatial patterns from a distributed hydrological model, remotely sensed surface temperatures, and sub-basin water balances. Hydrology and Earth System Sciences,17,7(2013-07-25), 2018, 17(7):2947-2966.*

*[5] Kargarian A, Mohammadi J, Guo J, et al. Toward Distributed/Decentralized DC Optimal Power Flow Implementation in Future Electric Power Systems. IEEE Transactions on Smart Grid, 2018, 9(4):2574-2594.*

*[6] Fioretto F, Pontelli E, Yeoh W. Distributed Constraint Optimization Problems and Applications: A Survey. Journal of Artificial Intelligence Research, 2018, 61(45):1-16.*

*[7] Margellos K, Falsone A, Garatti S, et al. Distributed constrained optimization and consensus in uncertain networks via proximal minimization. IEEE Transactions on Automatic Control, 2018, 63(5):1372-1387.*

*[8] E Sevinç, A Coşar. An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries. Computer Journal, 2018, 54(5):717-725.*

*[9] Mallik G, Nataraj N, Raymond J P. Error estimates for the numerical approximation of a distributed optimal control problem governed by the von Kármán equations. Esaim*

*Mathematical Modelling & Numerical Analysis, 2018, 52(3):1137-1172.*

*[10] Krause S, Blume T, Cassidy N J. Investigating patterns and controls of groundwater up-welling in a lowland river by combining Fibre-optic Distributed Temperature Sensing with observations of vertical hydraulic gradients. Hydrology and Earth System Sciences,16,6(2012-06-29), 2018, 9(1):337-378.*

*[11] Ferraz R, Ferraz R, Rueda-Medina A C, et al. Genetic optimisation-based distributed energy resource allocation and recloser-fuse coordination. IET Generation Transmission & Distribution, 2020, 14(20):4501-4508.*

*[12] Hamdani S, Khan A W, Iltaf N, et al. Dynamic distributed trust management scheme for the Internet of Things. Turkish Journal of Electrical Engineering and Computer Sciences, 2020, 29(2):796-815.*

*[13] A A E B, A A G, B J L, et al. CD30 + lymphomatoid skin toxicity secondary to ipilimumab. JAAD Case Reports, 2020, 6( 4):251-253.*

*[14] Dehghanpour K, Yuan Y, Bu F, et al. Statistical Modeling of Networked Solar Resources for Assessing and Mitigating Risk of Interdependent Inverter Tripping Events in Distribution Grids. IEEE Transactions on Power Systems, 2020, PP(99):1-1.*

*[15] Consolaro A. Extensive orthodontically induced dental resorption: What to do?. Dental Press Journal of Orthodontics, 2020, 25(2):18-23.*

*[16] Avelar S A, Piumbini R M, Ompolo S, et al. Karyotypic description of the stingless bee Meliponaquinquefasciata Lepeletier, 1836 (Hymenoptera, Meliponini) with emphasis on the presence of B chromosomes. Comparative cytogenetics, 2020, 12(4):471-482.*

*[17] Segura A, Duque E, Rojas A, et al. Functional characterization of fatty acyl desaturase Fads2 and Elovl5 elongase in the Boddart's goggle‐eyed goby, Boleophthalmus boddarti (Gobiidae) suggest an incapacity for long‐chain polyunsaturated fatty acid biosynthesis. 2020, 6(4):416-423.*

*[18] Kasim K, Deepthi R S. Morphological, Pathological and Molecular Diversity of Colletotrichum capsici inciting Fruit Rot in Chilli (Capsicum annuum L.). Research Journal of Biotechnology, 2020, 12(4):14-21.*