Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010302
ISSN 2790-0916 Vol. 1, Issue 3: 9-16

Fault Handling Method of Distributed System Based on Large Scale Dynamic Programming

Constantinos Kokkinos[*]

Democritus University of Thrace, Greece

[*]*corresponding author*

Keywords: Large Scale Network, Dynamic Planning, Distributed System, System Fault Handling Method

Abstract: with the rapid improvement of Internet and information construction in all walks of life, the traditional centralized storage system is gradually replaced by the distributed storage system. Fault processing method based on distributed system is a research hotspot in the field of power system fault identification. The purpose of this paper is to integrate large-scale dynamic programming with distributed system fault handling methods. Through the research of large-scale dynamic programming and its algorithm and distributed system, the relevant fault handling methods are proposed. In the experiment, we construct the system execution flow and use the dynamic planning algorithm to study and analyze the usability and universality of the distributed system dynamic planning application.

1. Introduction

With the improvement of the times and the progress of information technology, the distributed power system is more stable and intelligent than before. However, the failure of the power system is still unavoidable, and the failure is more diverse and complex [1]. This requires us to continue to increase the research on fault handling methods of distributed power system, and fault identification is the key issue. In this paper, the research on fault handling method of distributed system based on large-scale dynamic programming is a breakthrough.

Distributed system is the main form of computer application with increasing scale and complexity.Beschastnikh I studied that distributed systems pose unique challenges to software developers. It may be difficult to understand the communication topology of the system and infer the concurrent activities of the system host. The standard method of analyzing system logs may be a tedious and complex process, including rebuilding system logs from logs of multiple hosts, coordinating time stamps between hosts with asynchronous clocks, and understanding what happened during the execution of log coding. In this paper, a new method is proposed to deal with the three tasks frequently executed in the process of distributed system execution analysis: to understand the relative order of events; Searching for a specific mode of interaction between hosts; As well as the similarities and differences of the recognition structures between paired executions [2]. Merceedi K J proposed that in the past few days, data and Internet have become more and more large and appear in big data. To solve these problems, there are many software frameworks to improve the performance of distributed systems. The software is used to provide sufficient data storage. One of the most useful software frameworks for leveraging data in distributed systems is Hadoop. The software creates machine clusters and formats the work between them. Through Hadoop, we can process, calculate and distribute each word in a large file, and know the number of times each word is affected. The research aims to effectively store a large number of data sets and transmit them to high bandwidth user applications [3].On the one hand, the continuous improvement of system scale provides strong support for massive data storage and calculation, and on the other hand, it brings great challenges to the operation, maintenance and monitoring of business systems.

This paper studies large-scale dynamic programming and its algorithm, including dynamic programming and large-scale network clustering algorithm, introduces distributed system and analyzes the fault handling method of distributed power system based on large-scale dynamic programming. In the experiment, the system execution flow is constructed and the dynamic planning algorithm is used to study and analyze the usability of the distributed system dynamic planning application and the universality of the distributed system dynamic planning application.

2. Research on Fault Handling Method of Distributed System Based on Large Scale Dynamic Programming

2.1. Large Scale Dynamic Programming and its Algorithm

(1) Dynamic programming

Dynamic programming is a mathematical method for solving the optimization problem of multi-stage decision-making process [4-5]. According to the characteristics of a class of optimization problems of multi-stage decision-making process, the optimization principle for solving such problems is proposed. The multi-stage decision-making optimization problem to be solved is transformed into several sub stage problems, and each sub stage problem is solved by using the mutual relationship of each stage. Many practical problems and mathematical models are studied, thus a new branch of mathematical programming called dynamic programming is established [6-7].

(2) Clustering algorithm for large scale networks

As the network scale continues to increase, the disadvantages of clustering algorithms that can only deal with small-scale networks are prominent, and solving the scalability problem of these algorithms has become an important topic. In addition, the acceleration of network evolution requires more and more real-time graph clustering algorithms. Therefore, in recent years, more and more scholars have paid attention to the research of large-scale network clustering algorithms [8-9]. At present, there are two main ways to solve the large-scale network clustering algorithm. One is to reduce the scale of the network by removing nodes or edges; The second is to accelerate the algorithm by optimizing the clustering algorithm [10-11].

2.2. Distributed System

The distributed system is built on the computer network. The presentation layer, application layer, logic processing layer, computing layer and data layer of the application software system are logically and physically designed and distributed into a network. At the same time, the overall system and each application node have the characteristics of high cohesion and high transparency possessed by the software system [12-13]. Because of this characteristic of software, the distributed system is different from the traditional centralized system in structure, function and working mode. The distributed system has a high degree of cohesion, which is achieved in functions and modules. Each application node in the system is highly autonomous and managed. These application nodes work together under the support of the distributed system.

The distributed system is highly transparent, and can dynamically allocate tasks to each application node, so as to realize the reasonable allocation of physical and logical resources distributed among each application node. The distributed system is a complete and transparent whole for the user's application system. The user does not need to know which application node the task is executed on and which database node the data is saved in. The distributed architecture is adopted and the original working mode of the user will not be changed [14-15]. Distributed systems are gradually applied to more and more different types of applications, such as the following system applications. Using distributed architecture has more advantages than other centralized system architectures: high concurrent high-performance applications and high error tolerant applications.

2.3. Fault Handling Method of Distributed Power System Integrating Large-scale Dynamic Planning

There are two kinds of fault signal processing methods in distributed power system: signal based time domain analysis and signal based frequency domain analysis. The so-called signal based time-domain analysis here refers to the results obtained by processing and analyzing the time-domain signal sequence generated after the fault. It is the most direct and simple analysis method. If the signal contains a large number of periodic components or simple harmonic components, it is more effective [16-17]. The time domain analysis methods of signals mainly include time domain average method, correlation analysis diagnosis method and time domain waveform characteristic method. Frequency domain analysis is based on Fourier transform, which expands the time domain signal with Fourier series to obtain information of different frequency bands. It includes spectrum analysis and envelope analysis. Spectrum analysis can provide more intuitive and easy to distinguish feature information, and its feature information can also be used for signal trend estimation; Envelope analysis is mainly to extract the low-frequency signal in the signal. From the time domain, this method is to obtain the envelope trajectory curve of the time-domain waveform of the signal. Although these traditional signal processing technologies have been relatively mature and reliable in theory and practical application, but they have great limitations, so they are only applicable to the analysis of some stationary signals, and have no obvious effect on non-stationary signals [18-19]. In order to obtain the fault feature quantity of the system more effectively, more accurately and more conveniently, it requires us to analyze the signal from a new and more comprehensive perspective. Therefore, the fault information feature quantity extraction method based on non-stationary signal analysis deserves our in-depth study.

3. Investigation and Research on Fault Handling Method of Distributed System Based on Large Scale Dynamic Programming

3.1. Execution Process

The execution flow of dpx10 program is shown in Figure 1 without considering the occurrence of fault:

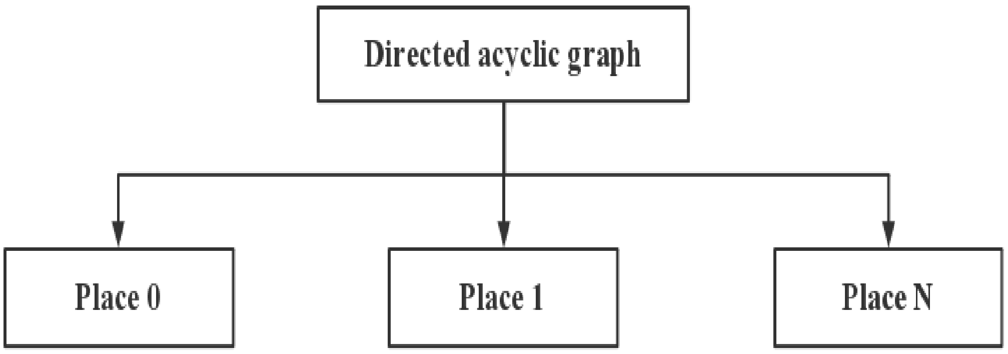

Figure 1. Running flow logic diagram of the DPX 10

3.2. Dynamic Programming Algorithm:

The specific formula of dynamic programming state planning algorithm is as follows:

Algorithm 1 (2D / 0d): given D [I, 0] and D [0, J], 1 ≤ I, J ≤ n,

$$D[i,j]=min\{D[i-1,j]+x_i,D[i,j-1]+y_i\} \tag{1}$$

x_i , y_i can be calculated in constant time.

Calculator 2(2D/1D):32473; defined w(i,j); 1≤i, j≤n; D(i)=0,1≤i,

$$D[i,j]=w(i,j)+min_{i\le k\le j}\{D[i,k-1]+D[k,j]\} \tag{2}$$

4. Analysis and Research on Fault Handling Method of Distributed System Based on Large Scale Dynamic Programming

4.1. Ease of Use of Distributed System Dynamic Planning Application

One of the goals of dpx10 is to provide developers with an easy way to write efficient distributed dynamic planning applications. Therefore, this section uses the amount of code to evaluate the ease of improving a dynamic planning program using dpx10. We compare the amount of code written by using dpx10 and directly using X10 to write the same program. The preprocessing process, post-processing process, comments and blank lines are not recorded. The code amount of distributed dynamic planning application is shown in Table 1 and Figure 2:

Table 1. Code quantity of the four dynamic planning applications (lines)

Apply	X10(serial)	X10(distributed)	DPX10
SW	53	157	53
MI P	31	154	58
LPS	30	186	47
0/1KP	29	165	82

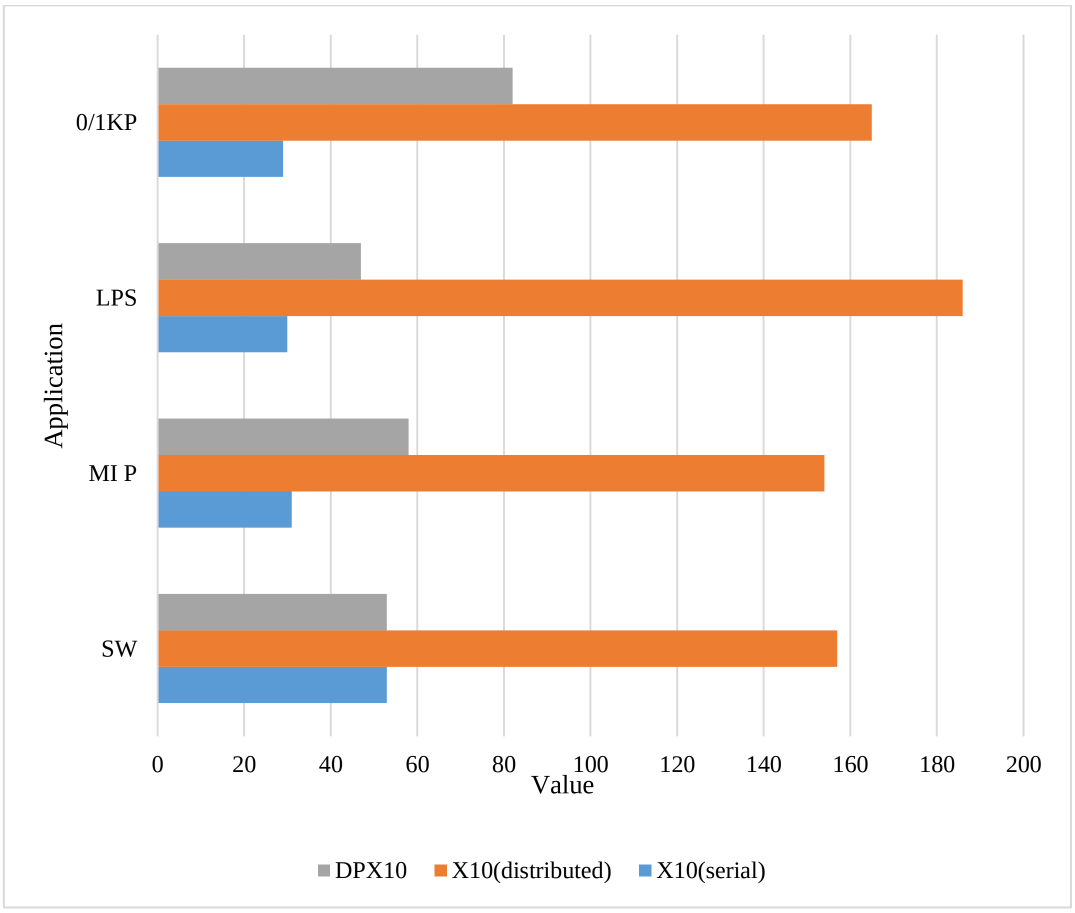

Figure 2. Dynamic planning and comparison diagram

The results show that the amount of code needed to write distributed programs directly using X10 is about four times that of serial programs. Moreover, in different distributed program codes, there are a large number of repetitive parts, such as distributed vertices and communication between workers. Dpx10 will try its best to hide these tasks from the user, so that the user can focus on the logic part of the algorithm.

4.2. Universality of Distributed System Dynamic Planning Application

Using the runtime of dpx10 fine-grained scheduler, dpx10 coarse-grained scheduler and z-align, the experiment was run on 15 computing nodes. Both z-align and dpx10 have good scalability. However, z-align is 2 times faster than coarse-grained dpx10 and about 9 times faster than fine-grained dpx10. The comparison of algorithm running time is shown in Table 2 and figure 3:

Table 2. DPX10 vs. Z-align runtime comparison

Vertex number	DPX10 (fine grain)	DPX10 (coarse particle size)	Z-align
200M	19.321	7.514	3.652
400M	33.564	8.654	4.652
600M	45.515	9.395	5.961
800M	56.324	11.698	6.654
1G	67.365	16.685	7.841

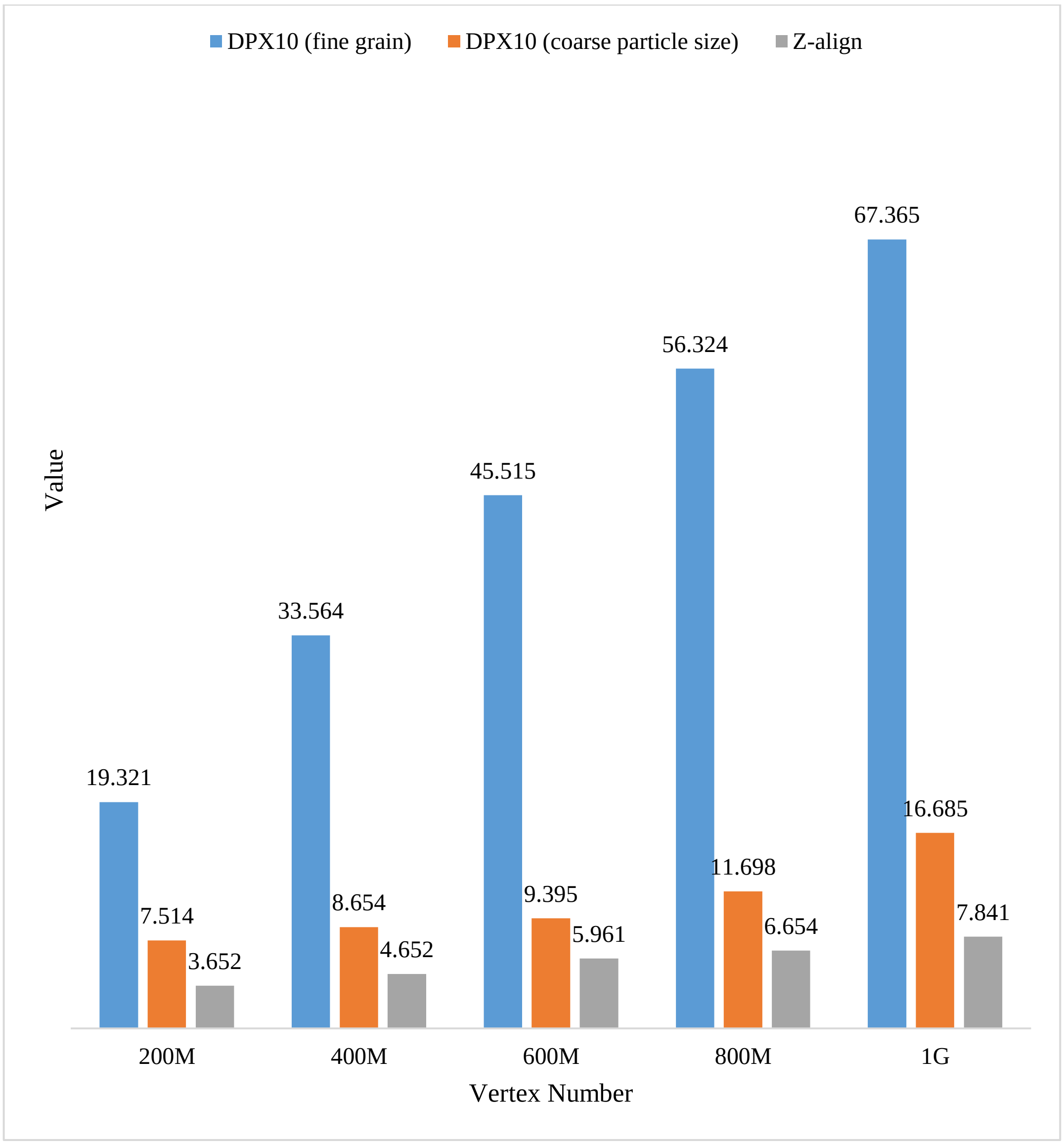

Figure 3. Algorithmic data comparison figure

The experimental results are acceptable because z-align is a specially optimized system, and dpx10 is a general system. Therefore, dpx10 must sacrifice part of its performance in exchange for the ease of use and versatility of the distributed system.

5. Conclusion

With the improvement of the times, the single computer system can not meet the needs of users in terms of computing speed and hardware resources, which promotes the rapid improvement of distributed systems; At the same time, with the popularization of distributed systems in the industry, the scale of distributed systems is expanding. This paper presents dpx10, a simple and powerful abstract model for dynamic programming type applications, to realize a distributed system integrating large-scale dynamic programming. Dpx10 uses both the shared memory model and the distributed memory model to make full use of the computing power of the distributed hardware environment. With the rapid improvement of large-scale distributed system, it has more and more requirements. Moreover, due to the increasingly wide application of distributed systems in the industry, the demand for mature distributed system monitoring technology will also increase. In the future work, we will consider integrating large-scale dynamic planning to support larger scale applications.

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] Ahmed M K, Ali A, Aminu A A, et al. Multi-Agent based Capital Market Management System: A Distributed Framework for Trading and Regulation. International Journal of Managing Information Technology, 2020, 13(2):1-14.

[2] Beschastnikh I, Liu P, Xing A, et al. Visualizing Distributed System Executions. ACM Transactions on Software Engineering and Methodology (TOSEM), 2020, 29(2):1-38.

[3] Merceedi K J, Sabry N A. A Comprehensive Survey for Hadoop Distributed File System. Asian Journal of Computer Science and Information Technology, 2020, 11(2):46-57.

[4] Fabris M, Michieletto G, Cenedese A. A Proximal Point Approach for Distributed System State Estimation. IFAC-PapersOnLine, 2020, 53(2):2702-2707.

[5] Tabasi M, Asgharian P. Optimal operation of energy storage units in distributed system using social spider optimization algorithm. AIMS Electronics and Electrical Engineering, 2019, 3(4):309-327.

[6] Tabasi M, Asgharian P. Optimal operation of energy storage units in distributed system using social spider optimization algorithm. International Journal on Electrical Engineering and Informatics, 2019, 11(3):564-579.

[7] Patil P, Bagwan T, Kulkarni S, et al. Multi-Attacks Detection in Distributed System using Machine Learning. International Journal Of Computer Sciences And Engineering, 2019,

7(1):601-605.

[8] Arif E. Implementing a Distributed System as a Solution to Problems in the Indosat Cooperative. JURNAL PETIK, 2019, 5(2):24-29.

[9] Widerski M, Gwd M. Solar Power Plant with Distributed System of PV Panels. Przeglad Elektrotechniczny, 2019, 95(2):55-58.

[10] Mingnan, Zhou, Dahai, et al. An Efficient Code-defect Testing Distributed System. IOP Conference Series: Materials Science and Engineering, 2019, 490(4):42038-42038.

[11] Seshadri G, Marutheswar G V. A Novel Architecture for Series-Connected Photovoltaic Distributed System with Enhanced Power Quality Using Type-II Fuzzy Controller. International Journal of Intelligent Engineering and Systems, 2018, 11(4):177-187.

[12] Basinya E A. Distributed system of collecting, processing and analysis of security information events of the enterprise network infrastructure. Bezopasnost Informacionnyh Tehnology, 2018, 25(4):43-52.

[13] Wuning, Tong, Song, et al. Fault-Tolerant Scheduling Algorithm with Re-allocation for Divisible Loads on Homogeneous Distributed System. IAENG Internaitonal journal of computer science, 2018, 45(3):450-457.

[14] Ahad M A, Biswas R. PPS-ADS: A Framework for Privacy-Preserved and Secured Distributed System Architecture for Handling Big Data. International Journal on Advanced Science Engineering and Information Technology, 2018, 8(4):1333-1342.

[15] Chowhan R S, Dayya P. Next Generation in Computing with Agent Oriented Distributed System: Protocols and Features. Oriental Journal of Computer Science & Technology, 2018, 11(2):126-134.

[16] Abhijit A, Sulabha S. Performance Enhancement of Distributed System through Load Balancing and Task Scheduling. International Journal of Computer Applications, 2018, 181(3):20-26.

[17] Fedorov A, Ovseevich A. Asymptotically optimal dry-friction like control for a simplest distributed system. IFAC-PapersOnLine, 2018, 51(32):87-92.

[18] Hanane C, Battou A, Baz O. Performance Security in Distributed System: Comparative Study. International Journal of Computer Applications, 2018, 179(15):29-33.

[19] Burdonov I B, Kossatchev A S. Directed distributed system: Backtracking problem. Proceedings of the Institute for System Programming of RAS, 2018, 30(2):167-194.