A Multi-Point Observation Data Storage and Synchronization Method Based on NGAS (Post-print)

Authors: Shi Congming, Deng Hui, Dai Wei, Shoulin Wei, Wang Feng

Date: 2018-09-11T00:00:00+00:00

Abstract

The Square Kilometre Array (SKA) telescope, upon completion, will possess ultra-high sensitivity, ultra-fast survey speed, and a wide field of view, thereby generating extremely massive observational data. The massive data synchronization/transfer between the SKA Observatory and national regional data centers is a current challenge in SKA construction. The Next Generation Archive System (NGAS) used in SKA pilot projects exhibits issues such as low efficiency and insufficient performance in application testing. This paper proposes a data storage and synchronization method based on ZeroMQ, which implements synchronous data transmission through a more efficient asynchronous messaging mechanism, circumventing the limitations of NGAS' s original HTTP protocol. Experimental results demonstrate that the new method is nearly 40 times faster than the original NGAS method in terms of average data archiving and storage efficiency, can essentially meet the full-speed transmission requirements of 10GB bandwidth, and has achieved satisfactory usage results.

Full Text

Preamble

NGAS-Based Multi-site Observation Data Storage and Synchronization Method

Shi Congming[13], Deng Hui[23], Dai Wei[34], Wei Shoulin[3], Wang Feng[1234]

[1]Faculty of Management and Economics, Kunming University of Science and Technology, Kunming 650093, China
[2]Center for Astrophysics/Institute of Physics and Electronic Engineering, Guangzhou University, Guangzhou 510006, China
[3]Key Laboratory of Applications of Computer Technology of Yunnan Province,

Kunming University of Science and Technology, Kunming 650504, China
[4]Yunnan Observatories, Chinese Academy of Sciences, Kunming 650216, China

Abstract

The Square Kilometre Array (SKA) will be the largest radio telescope ever constructed, featuring unprecedented sensitivity, survey speed, and field of view. Upon completion, it will generate super-massive volumes of observation data, presenting a major challenge for data transmission and archiving during the SKA construction phase. The Next Generation Archive System (NGAS), currently used by SKA precursor projects, exhibits inefficiency and insufficient performance for these demanding requirements. This paper proposes a more efficient data storage and synchronization method based on ZeroMQ, which implements synchronization through an asynchronous messaging mechanism that avoids the limitations of NGAS' s HTTP-based protocol. Experimental results demonstrate that the new method achieves nearly 40 times faster average data archival/storage efficiency compared to the original NGAS implementation, can essentially meet the requirements for full-speed transmission over 10GB bandwidth, and has achieved satisfactory practical results.

Keywords: NGAS; SKA; Storage and Synchronization; Massive Data; ZeroMQ

1. Analysis of NGAS Underlying Implementation

NGAS is a multi-threaded concurrent HTTP server at its core, utilizing a relational database management system (RDBMS) to manage metadata for archived files, subscriber information, and disk status. NGAS implements over 20 custom commands including STATUS, ONLINE, OFFLINE, ARCHIVE, SUBSCRIBE, and UNSUBSCRIBE. These commands primarily support fundamental functions such as data archiving and retrieval, server-side data compression and filtering, automatic data mirroring, disk tracking, offline data transfer, data consistency verification, and data subscription (storage and synchronization).

1.1 NGAS Data Synchronization Functionality

NGAS's data synchronization transmission capability is achieved through NGAS data subscription threads that schedule corresponding data sending threads for each subscriber. The flowchart for the data sending thread is shown in Figure 1 [Figure 1: see original paper].

Each data file transmission from an NGAS publisher to a subscriber involves the following process: the data file is encapsulated using the HTTP protocol, the encapsulated file is transmitted, the publisher waits to receive a successful storage confirmation message from the subscriber, and only then proceeds to the next file. This requirement for the publisher to wait for subscriber feedback during synchronous data transmission, combined with the low encapsulation efficiency of HTTP protocol, results in poor overall data transmission performance.

2. Improvements to Multi-site Observation Data Storage and Synchronization Methods

To address NGAS' s HTTP-based data synchronization limitations and align with current technological trends, this paper proposes an improved method for multi-site observation data storage and synchronization in NGAS using ZeroMQ. The improved approach employs ZeroMQ's PUB-SUB socket pattern for efficient and rapid data synchronization transmission and storage. However, the PUB-SUB pattern presents several challenges: (1) subscriber crashes causing data loss; (2) slow subscriber message retrieval leading to publisher queue overflow and data loss; (3) network overload causing data loss; and (4) late-joining subscribers missing previously published data.

To resolve these issues in our improved implementation, we incorporated a near-real-time port connection status monitoring mechanism to prevent data publication when no subscribers are connected, along with a data retransmission mechanism to overcome data loss caused by network overload or subscriber crashes, ensuring complete data synchronization. To enable the improved data synchronization subsystem to operate independently of NGAS, we integrated subscription and unsubscription functionality modules using ZeroMQ' s DEALER and ROUTER sockets.

The system implemented based on this improved method comprises four main components: Publisher Server (Pub-Server), Subscriber Server (Sub-Server), Subscription Management Server (Subscriber-Server), and Subscription Client (Subscriber-Client). Pub-Server and Sub-Server handle data synchronization transmission and storage between publishers and subscribers, as shown in Figure 2 [Figure 2: see original paper]. Subscriber-Server and Subscriber-Client manage message subscription and unsubscription between publishers and subscribers, as shown in Figure 3 [Figure 3: see original paper].

2.1 Design and Implementation of Pub-Server and Sub-Server

Pub-Server is primarily responsible for launching daemon threads related to data publishing and synchronization transmission at the publisher side. Its execution flowchart is shown in Figure 4 [Figure 4: see original paper]. Pub-Server includes the following functional modules: launching publisher-specific data publishing daemon threads, receiving feedback message daemon threads, processing feedback message daemon threads, updating backlog file daemon threads, updating publication queue daemon threads, processing new subscriber daemon threads, and processing new unsubscription daemon threads. Sub-Server functions similarly but operates on different objects.

Data synchronization transmission and storage between Pub-Server and Sub-Server involves two message types: Pub_{msg} and Sub_{msg}, as illustrated in Figure 2. Pub_{msg} is the message published by the data publisher, with the format `SI_{SP}:PI_{PP}:BFR:BFD`. Sub_{msg} is the feedback information published by the data subscriber upon successful receipt and storage, with the

format SI_{SP}:PI_{PP}:BFR. Here, SI represents the Subscriber' s IP address; SP is the fixed port assigned by the Subscriber for receiving feedback messages from a specific Publisher; PI represents the Publisher' s IP address; PP is the fixed port reserved by the Publisher for publishing data to a specific Subscriber; BFR consists of filename, file ID, file version, and file type; and BFD represents the backlog file data corresponding to BFR.

When Pub-Server's near-real-time port connection status daemon thread detects that a Publisher' s data publication port has been connected by a Subscriber, it triggers the corresponding data publishing thread to begin generating and publishing Pub_{msg}; otherwise, it stops the data publishing thread. Simultaneously, the data retransmission mechanism ensures that if a Publisher does not receive a successful receipt and storage feedback message from a Subscriber within a specified timeout period after publishing a data file, the Publisher will retransmit that data file to the Subscriber.

2.2 Design and Implementation of Subscriber-Server and Subscriber-Client

The asynchronous communication pattern between Subscriber-Server and Subscriber-Client is shown in Figure 3 [Figure 3: see original paper]. Subscriber-Server receives and processes subscription messages (Sub-Msg) and unsubscription messages (Unsub-Msg), replies with subscription success messages (Sub-Msg-S), subscription failure messages (Sub-Msg-F), or unsubscription success messages (Unsub-Msg-S) to the requesters, and updates corresponding subscriber records in the database. Subscriber-Client sends Sub-Msg and Unsub-Msg to Subscriber-Server and updates corresponding publisher records in the database based on received response messages.

The five message formats are: S_{{{SI}}_{{SP}}}_{Datetime}, U_{{{SI}}_{{SP}}}, SS_{{{SI}}_{{SP}}}_{{{PI}}_{{PP}}}, SF_{{{SI}}_{{SP}}}, and US_{{{SI}}_{{SP}}}. Additionally, we incorporated a message retransmission mechanism in Subscriber-Client to ensure successful subscription or unsubscription to corresponding data sources.

3. Experimental Evaluation

3.1 Experimental Environment

The hardware environment for performance testing consisted of one IW4200-10G GPU server from SITONHOLY equipped with 16 dual-core Intel® Xeon(R) CPU E5-2620 v4 @ 2.10GHz processors, 256GB R-ECC DDR4 memory, and two Intel® I350 gigabit network cards. The software environment comprised 64-bit Ubuntu 14.04 LTS, Python 2.7.6, MySQLdb 1.2.5, libzmq 4.2.5, pyzmq 17.1.2, and MySQL 5.5.61.

Since NGAS can only process standard FITS files containing the custom keyword ARCFILE (e.g., ARCFILE = 'NCU.2003-11-11T11:11:11.111'), our exper-

imental dataset consisted of 400,000 FITS files from MUSER-I with the ARCFILE keyword added, totaling approximately 75.102GB (400,000 × 201,600B), stored in a tmpfs (temporary file system) allocated with 250GB of memory.

3.2 Experimental Results

The performance comparison between the ZeroMQ-improved multi-site observation data storage and synchronization method and the original NGAS method is shown in Figure 5 [Figure 5: see original paper]. Synchronizing and storing all 400,000 FITS files using the ZeroMQ-based method required approximately 333.834 seconds (about 5.6 minutes), whereas the NGAS method required approximately 13,330.998 seconds (about 222.2 minutes). The ZeroMQ-based method achieved a speedup of approximately 39.993 times over the NGAS method.

Experimental results demonstrate that the ZeroMQ-improved NGAS data storage and synchronization method significantly outperforms the original NGAS approach. However, the method has some limitations: (1) Since the ZeroMQ-based NGAS publisher server must allocate a fixed port for each subscriber and is limited by the maximum of 65,536 ports per IP address, it can only serve a limited number of subscribers; (2) The subscriber server may be terminated due to memory exhaustion when it cannot store high-speed incoming subscription data promptly. Future work will incorporate dynamic data publication adjustment mechanisms to optimize synchronization transmission efficiency.

4. Conclusion

This paper has presented a detailed analysis of NGAS' s data synchronization functionality and discussed the ZeroMQ-based improved method for NGAS multi-site observation data storage and synchronization. Experimental validation demonstrates that the system implemented using this improved method significantly outperforms the original NGAS in data synchronization transmission and storage efficiency. Future work will test the remote data synchronization performance of the new method in more realistic experimental environments and further optimize its performance.

This research provides valuable reference for data synchronization transmission and storage between SKA regional data centers and SKA observatory data centers.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFA0404603, 2016YFE0100300), the Key Program of the Joint Fund for Astronomy of the National Natural Science Foundation of China (NSFC) and Chinese Academy of Sciences (U1831204), the Joint Fund for Astronomy of NSFC and Chinese Academy of Sciences (U1531132, U1631129, U1831204),

NSFC projects (11403009, 11463003, 11773012), Guangzhou University "Innovation Strengthening University" Project (2017KZDXM062), Yunnan Provincial Applied Basic Research Project (2017FB001), and CERNET Next Generation Internet Technology Innovation Project (NGII20170204). We thank the National Astronomical Observatories-Alibaba Cloud Astronomy Big Data Joint Research Center for supporting this work.

References

[1] Peng Bo, Jin Chengjing, Du Biao, et al. The world' s largest integrated aperture radio telescope SKA collaboration [J]. Bulletin of Chinese Academy of Sciences, 2011, 26(5): 487-495.

[2] Yan Jun. Strategic considerations on astrophysics research and future development in China [J]. Scientia Sinica: Physica, Mechanica & Astronomica, 2013, 42(12): 1292-1307.

[3] Broekema P C, Van Nieuwpoort R V, Bal H E. The Square Kilometre Array science data processor. Preliminary compute platform design [J]. Journal of Instrumentation, 2015, 10(07): C07004.

[4] Dewdney P E, Hall P J, Schilizzi R T, et al. The square kilometre array [J]. Proceedings of the IEEE, 2009, 97(8): 1482-1496.

[5] Hall P, Schillizzi R, Dewdney P, et al. The square kilometer array (SKA) radio telescope: Progress and technical directions [J]. International Union of Radio Science URSI, 2008, 236: 4-19.

[6] Braun R, Keane E, Bourke T, et al. Advancing Astrophysics with the Square Kilometre Array [J]. PoS, 2015, 174.

[7] Wicenec A, Knudstrup J, Johnston S. ESO' s Next Generation Archive System [J]. The Messenger, 2001, 106: 11-13.

[8] Wu C, Wicenec A, Pallot D, et al. Optimising NGAS for the MWA Archive [J]. Experimental Astronomy, 2013, 36(3): 679-694.

[9] Wicenec A, Knudstrup J. ESO' s next generation archive system in full operation [J]. The Messenger, 2007, 129: 27-31.

[10] Wicenec A, Chen A, Checcucci A, et al. The ALMA Front-end Archive Setup and Performance; proceedings of the Astronomical Data Analysis Software and Systems XIX, F, 2010 [C].

[11] Stoehr F, Lacy M, Leon S, et al. The ALMA archive and its place in the astronomy of the future; proceedings of the Observatory Operations: Strategies, Processes, and Systems V, F, 2014 [C]. International Society for Optics and Photonics.

[12] Harrison P, Knudstrup J, Wicenec A, et al. Implementing a VOStore Interface for NGAS; proceedings of the Astronomical Data Analysis Software and

Systems XV, F, 2006 [C].

[13] Hintjens P. ZeroMQ: messaging for many applications [M]. O' Reilly Media, Inc., 2013.

[14] Hintjens P. Ømq-the guide [J]. Online: http://zguide.zeromq.org/page:all, Accessed on, 2011, 23.

Note: Figure translations are in progress. See original paper for figures.