Data Processing Methods for High-Speed Payload Data Streams (postprint)

Authors: Wang Jing, Wang Chunmei, Zhijia, Jiasen Yang, Chen Tuo

Date: 2017-03-10T00:00:00+00:00

Abstract

In response to the characteristics of satellite payload data transmission, including high data rates and challenges in real-time processing, this paper proposes a real-time data processing method oriented toward high-speed payload data streams. The method adopts the multi-threaded parallel paradigm of MapReduce, combining hash algorithms with merge sort algorithms to enhance data processing throughput and achieve real-time processing; it employs a parameter parsing algorithm based on the XTCE (XML Telemetry & Command Exchange) data model to ensure universality. Experimental results demonstrate that the proposed method satisfies the requirements for both real-time performance and correctness in payload data processing.

Full Text

Data Processing Method for High-Speed Payload Data Streams

WANG Jing[1],[2], WANG Chun-Mei[1], ZHI Jia[1], YANG Jia-Sen[1], CHEN Tuo[1] [1]National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

Aiming at the characteristics of high transmission rates and difficult real-time processing for satellite payload telemetry data, this paper proposes a real-time data processing method for high-speed payload data streams. Drawing on the multi-threaded parallel pattern of MapReduce, the method combines hash algorithms with merge sort algorithms to improve data processing throughput and achieve real-time processing. Additionally, a parameter parsing algorithm based on the XTCE (XML Telemetry & Command Exchange) data model is

employed to ensure generality. Experimental results demonstrate that the proposed method can meet payload requirements for both real-time performance and data validity.

Keywords: high-speed payload data stream; data processing; MapReduce; throughput rate; XTCE

Author Biographies:
WANG Jing (1990–), female, from Weihai, Shandong, Ph.D. candidate, research interests include space data processing and automatic test technology.
WANG Chun-Mei (1965–), female, from Beijing, professor and Ph.D. supervisor, research interests include automatic test technology.
ZHI Jia (1984–), male, from Taiyuan, Shanxi, engineer, M.S., research interests include space data processing.
YANG Jia-Sen (1979–), male, from Liaocheng, Shandong, associate researcher, Ph.D. candidate, research interests include automatic test technology.
CHEN Tuo (1986–), male, from Tianmen, Hubei, engineer, M.S., research interests include database technology.
E-mail: wangj711@yeah.net

0 Introduction

With the rapid development of China' s aerospace industry, space science missions are generating increasingly large volumes of payload telemetry data with more complex data types. Meanwhile, continuous optimization of communication equipment performance has led to substantial improvements in data transmission rates. For high-speed payload data streams, even with increasingly powerful computer processing capabilities, conventional frame-by-frame parameter parsing methods still cannot meet science mission requirements for real-time performance and data validity. Consequently, research on data processing methods for high-speed payload data streams has gradually become a critical challenge in aerospace data processing.

Current research on large-scale data batch processing technologies is relatively mature both domestically and internationally, with the MapReduce framework widely applied for batch processing computations. However, existing large-scale data processing technologies such as Hadoop and Phoenix are more oriented toward handling persistent data and cannot satisfy the real-time performance and throughput requirements of big data stream computing. Moreover, these technologies face compatibility issues with existing data processing modules. Stream computing research in China remains in its infancy, while foreign systems such as Twitter' s Storm and Yahoo' s S4 have been developed but are more suitable for data mining applications with lower requirements for data processing correctness and real-time performance.

Payload data processing emphasizes a series of operations with high correctness requirements, including synchronization code recognition, data packetization and reassembly, parameter data extraction, physical quantity conversion, data

calibration, correction, and inversion. These requirements differ significantly from existing stream computing system applications. Furthermore, multi-model and multi-mission operations represent the future direction of aerospace development, necessitating data processing methods with good generality to accommodate heterogeneous missions and effectively reduce development costs. In light of these considerations, this paper designs a generic data processing method suitable for high-speed payload data streams.

1 Payload Real-Time Data Processing System

Payload telemetry data, referred to as transmission frames, follows the Advanced Orbiting Systems (AOS) standard of the Consultative Committee for Space Data Systems (CCSDS). An example structure is shown in . The data processing procedure is complex and primarily consists of two parts: preprocessing and parameter parsing, typically supplemented by a parameter storage system for data preservation and a data monitoring system for supervising the status of each component. The system structure is illustrated in [Figure 1: see original paper].

The preprocessing module performs packetization and assembly processing on the data domain of transmission frames to form the data frame structure shown in . This processing is relatively fast and generally does not create rate bottlenecks. The parameter parsing module processes each parameter in the data frame. The application data segment length in data frames is variable; in engineering applications, there are dozens of application data formats, each containing tens to hundreds of parameters. More parameters and more complex value conversions result in more time-consuming parsing processes.

With the development of space science missions, payload data volume has increased significantly, substantially raising requirements for data processing throughput. Due to the complexity and time-consuming nature of parameter parsing modules, conventional frame-by-frame processing methods cause severe data accumulation when handling high-speed payload data streams, degrading real-time performance and correctness and even causing system failures. To address this problem, this paper designs a data processing method for high-speed payload data streams that draws on the divide-and-conquer philosophy of the MapReduce model. By designing a multi-threaded parallel pattern combining hash algorithms with merge sort algorithms, the method achieves real-time data processing for high-speed payload data streams, meeting space science mission requirements for throughput, real-time performance, and correctness.

2 Key Technologies

The design of the processing method for high-speed payload data streams focuses on improving parameter parsing module performance through two aspects: first, adopting an efficient processing pattern that changes from frame-by-frame to parallel processing without affecting data correctness; second, employing a

generic parsing algorithm capable of supporting multi-mission scenarios with good versatility.

2.1 Parallel Processing Pattern

MapReduce batch processing methods provide sufficient parallel computing semantics for divisible large-scale data processing tasks. For high-speed payload data streams, this paper adopts a multi-threaded parallel processing pattern inspired by the divide-and-conquer philosophy. The data processing architecture is shown in [Figure 2: see original paper] and can be described in three steps:

(1) Hash Shunting
By recognizing synchronization codes, the data stream is segmented into data frames. Using independent data frames as objects, hash methods perform equivalence mapping and allocate frames to corresponding hash buckets. Common hash function construction methods include:

- Direct remainder method: $\text{Hash}(x) = x \mod p$, where p is the number of hash buckets (or hash linked list length).
- Multiplication remainder method: $\text{Hash}(x) = \lfloor m \times (x \times \theta \mod 1) \rfloor$, where $0 \leq \theta \leq 1$.
- Square middle method: Take the middle digits after squaring.

To achieve true multi-threaded parallelism, the number of hash buckets for processing high-speed payload data streams should be less than twice the number of computer processor cores. Space data processing computers are typically high-performance systems with 32 or more cores, so the number of hash buckets can generally be selected from several dozen to several tens. The direct remainder method is simple to implement, ensures uniform data distribution, and represents an optimal choice.

(2) Data Frame Processing
This step implements parameter parsing for data frames. Each hash bucket storing data frames is allocated a dedicated processing thread and result buffer. Processing threads retrieve data frames from their buckets in FIFO (First Input First Output) order, parse them into corresponding parameters according to data frame format specifications, and finally store the parameter contents in their respective result buffers.

(3) Merging
Parameter results from each result buffer are merged in chronological order and sent to the parameter storage software. Due to the special design of this method, the sorting process can be implemented without comparing time fields by simply taking one parameter frame sequentially from result buffers 1 through n, which naturally yields chronological order. Each result buffer also adopts a FIFO queue structure.

In [Figure 2: see original paper], the data reception buffer, data frame buffer, and processing result buffer are all designed as circular queue structures, enabling

repeated use of fixed-size memory space without requiring dynamic memory allocation and deallocation.

2.2 Data Frame Processing

Processing data frames requires structural information about the frames and their parameters, while parsing results must also satisfy user display requirements. Traditionally, data frame processing required establishing separate databases and developing specific parameter parsing software for each mission' s data characteristics, resulting in heavy workload and long development cycles, particularly failing to meet future multi-satellite, multi-mission aerospace requirements. Introducing a standardized data information description model enables parameter parsing modules to transcend single aerospace missions and achieve generality.

CCSDS has introduced XTCE (XML Telemetry & Command Exchange) as an international standard for describing data models, presented in the form of XML schema files with the top-level root node SpaceSystem containing six components: Header, Description, SpaceSystem, CommandMetaData, TelemetryMetaData, and ServiceSet. The structural information required by parameter parsing modules primarily originates from the TelemetryMetaData element, whose child elements include ParameterTypeSet, ParameterSet, and ContainerSet.

XTCE has broad coverage but can be tailored to specific requirements to avoid redundancy. For parameter parsing modules, information such as parameter names, start/end positions, types, and conversion formulas is typically required. During system initialization, XTCE files are read, parsed, and saved as data models for use by each data frame processing thread. The algorithm flow for data frame processing threads is shown in [Figure 2: see original paper]. In the initial conditions, the variables represent:

- IsSyn: synchronization status
- MoveBit: required bit shift amount
- LenofUnPrecess: length of unprocessed data
- LenofLeft: length of remaining data from previous processing
- UnitDataLength: unit processing length (data frame length)

The core algorithm concept involves setting up two buffers, each sized at twice the maximum length for a single processing operation. Buffer B1 is used for processing, while buffer B2 serves as backup to restore data in B1 when synchronization code searching causes shifting. Newly read data LenofNewData should not exceed the maximum single-processing length. Fixed-length data is read each time to check synchronization status; if synchronized, parsing proceeds. If not synchronized, synchronization codes are re-located and parsing begins from the new position. If no synchronization code is found, only data at the end of the current processing window equal to the synchronization code length is retained to ensure subsequent synchronization state identification is not

affected. Synchronization code searching uses byte-by-byte comparison; if not found in one comparison, buffer B1 is right-shifted by one bit for re-comparison, with a maximum of 7 shifts.

This algorithm performs parameter parsing by matching data models in XTCE files, not limited to single structures and demonstrating good generality.

3 System Implementation and Verification

A data processing system for high-speed payload data streams was developed using Visual Studio 2012 and deployed on a Linux server. The system was tested using telemetry data from a specific satellite model requiring real-time processing throughput no less than 150 Mbps. The test verification platform is shown in [Figure 3: see original paper]. A simulation transmission program was installed on the transmission simulation computer to send telemetry data to the data processing server via network at specified rates. The data processing system was deployed on the server, with all modules (including the database) interconnected via 10 Gigabit Ethernet. System status information was sent to the data processing monitoring system via Gigabit Ethernet.

Testing demonstrated that the proposed method for high-speed payload data streams outperforms conventional methods, with comparative results shown in . It should be noted that the conventional method uses post-processing; when employed for real-time processing, rates exceeding 12.1 Mbps cause receive buffer overflow and processing interruption.

Additionally, the generic parameter parsing algorithm enables the parameter parsing module to support multi-mission scenarios. When data structures change, only the XTCE file needs updating without program modifications, significantly reducing development costs.

Table 3 Comparison of Test Results Between Two Methods

Method	Single-thread Frame-by-Frame Mode	Multi-thread Parallel Mode
Server Configuration	16 cores, 64GB memory	16 cores, 64GB memory
Simulated Transmission Rate	12.1 Mbps	145 Mbps, 155 Mbps
Maximum Throughput	>155 Mbps (16 threads)	

4 Conclusion

This paper designs and implements a data processing system for high-speed payload data streams that addresses the inability of conventional methods to meet high data processing throughput requirements while providing good generality for multi-model, multi-mission scenarios. System testing and verification demonstrate that the proposed solution can meet current processing requirements for high-speed payload data streams, providing a valuable reference for engineering applications.

References

[1] WANG Yuanzhuo, JIN Xiaolong, CHENG Xueqi. Network big data: Present and future[J]. Chinese Journal of Computers, 2013, 36(6):1125-1138.

[2] YANG D, RUNDENSTEINER EA, WARD M. Mining neighbor-based patterns in data streams[J]. Information Systems, 2013, 38(3):331-350.

[3] LIM L, MISRA A, MO TL. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams[J]. Distributed and Parallel Databases, 2013, 31(2):321-351.

[4] QI Kaiyuan, ZHAO Zhuofeng, FANG Jun, et al. Real-time processing for high speed data stream over large scale data[J]. Chinese Journal of Computers, 2013, 35(3):477-490.

[5] SUN Dawei, ZHANG Guangyan, ZHENG Weimin. Big stream computing: Key technologies and system instances[J]. Journal of Software, 2014, 25(4):839-862.

[6] Recommended Standard CCSDS 732.0-B-3. AOS SPACE DATA LINK PROTOCOL[S].

[7] ANAND R, JEFFREY D U. Mining of massive datasets[M]. Beijing: The People' s Posts and Telecommunications Press, 2012.

[8] DEAN J, GHEMAWAT S. MapReduce: Simplified data processing on large clusters[J]. ACM Communication, 2008, 51(1):107-113.

[9] LI Yulin, DONG Jing. Research and improvement of MapReduce model based on Hadoop[J]. Computer Engineering and Design, 2012, 33(8):3110-3116.

[10] YAN Weimin, WU Weimin. Data structure (C language)[M]. Beijing: Tsinghua University Press, 2006:253-362.

[11] Recommended Standard CCSDS 660.0-B-1. XML TELEMETRIC AND COMMAND EXCHANGE (XTCE)[S]. 2007.10.

[12] QU Yi, LIU Yurong, ZUO Jiangtao, et al. Study on the architecture of telemetry processing software based on XTCE[J]. Journal of Spacecraft TT&C Technology, 2012, 31(1):60-64.

[13] CAO Yujuan, WANG Jun, OU Yujun, et al. Review of CCSDS XTCE[J]. Journal of Spacecraft TT&C Technology, 2012, 31(suppl.):38-42.

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