Distributed AI embedded cluster for real-time video analysis systems with edge computing

Wei Li*, *Zhiyuan* Han, *Jian* Shen, *Dandan* Luo, *Bo* Gao and *Jin* Xie

Insigma Technology Co. Ltd., Hangzhou, China

Abstract. Herein, on the basis of a distributed AI cluster, a real-time video analysis system is proposed for edge computing. With ARM cluster server as the hardware platform, a distributed software platform is constructed. The system is characterized by flexible expansion, flexible deployment, data security, and network bandwidth efficiency, which makes it suited to edge computing scenarios. According to the measurement data, the system is effective in increasing the speed of AI calculation by over 20 times in comparison with the embedded single board and achieving the calculation effect that matches GPU. Therefore, it is considered suited to the application in heavy computing power such as real-time AI computing.

Keywords: Distributed system, Embedded cluster, AI, Video analysis, Edge computing.

1 Introduction

According to an analysis of the annual shipments of surveillance cameras over the last 15 years, IHS Markit | Technology projects that there have been 770 million cameras put in service globally by the end of 2019. The number of cameras installed will reach above 1 billion over the next two years[1]. Based on facial and vehicle algorithms, AI has been applied extensively in various sectors, for example, public security, transportation and more. A new focus of research direction is how artificial intelligence can be applied to the intelligent transportation industry. Currently, the results are artificially judged using various sensor and video data for the bridge, island and tunnel to be maintained. As the bridge industry develops rapidly, a variety of sensors have been rapidly deployed, which makes it extremely difficult to achieve artificial recognition. As AI technology advances, particularly the advancement of computer vision technology and neural network technology, a possibility has been opened up for AI to support engineers in the research and judgment of various sensor data and video surveillance of bridges. In the meantime, however, there are problems arising from the practice of applying the cloud AI model in bridge scenarios, for example, data privacy, network bandwidth constraint, and demanding requirements on latency.

*Corresponding author: liwei@watone.com.cn

Most recently, AI chips based on the ARM architecture have attracted increasing attention. Such modules as CPU, GPU, ISP, and AI acceleration engine can be integrated by these purpose-built processors for AI computing. A single chip is capable of providing up to 195.7 TOPS/W of AI computing power at maximum[2]. As compared to general-purpose processors such as GPU, the chip of this type is advantageous for low power consumption, the simplicity of deployment, and excellent real-time performance, which makes it suited to edge computing. In high computing power requirement cases, such as real-time computing, however, the requirements can't be satisfied by one single ARM chip. For instance, there are as many as 524 surveillance cameras across the whole bridge in the transportation video analysis on the Hong Kong-Zhuhai-Macao Bridge. For mitigating the loss of bandwidth resource and ensuring data security, it is essential to deploy AI computing services at the edge, and diversified scenario puts forward higher requirements on the computing power performance of the device. In order to apply various AI algorithms such as pedestrian detection, obstacle detection, sign detection, and vehicle trajectory tracking, the use of distributed embedded AI cluster is proposed in this paper to connect the video stream of the existing surveillance cameras of the bridge. A two-layer architecture design of edge computing and central computing is adopted for the distributed AI platform. Edge computing contributes to the capability of processing data at the bottom and highly scalable distributed capabilities, while central computing leads to a sensible application architecture based on the business layer. In doing so, not only is the underlying computing and application system with "strongly distributed and loosely coupled" aligned with the developmental trend of information technology, it also satisfies the demands for dynamic upgrades in future applications.

2 Hardware platform introduction

Herein, Rockchip RK3399 core board is taken as the carrier of hardware computing. With a maximum clock speed of 1.8 GHz, the CPU of RK3399 is capable of integrating dual-core(Cortex-A72) and quad-core(Cortex-A53). In the meantime, RK3399 is able to support the decoding of 4K 60fps H.264/H.265/VP9 video hardware. Fitted with a total of eighty RK3399 core boards, a single distributed AI cluster has dual power supply (one main, one standby). The full load power consumption is 500W. Fig. 1 illustrates the overall design of the chassis in 3u specifications, which is complete with 6 hydraulic bearing cooling fans.

3 Distributed computing system architecture

3.1 Brief introduction of distributed system

The system is comprised of two components: a central server and an ARM cluster server. The former represents the core node of the entire architecture. Not only is it involved in task distribution and scheduling, as well as node service status monitoring, it is also intended to visualize data through the provision of user interface. A single ARM cluster server is made up of 80 channels consisting of RK3399 embedded core boards. One single node is capable of completing such tasks as video stream decoding and AI calculation. In addition, it is involved in reporting service status and keeping track of the tasks performed by other nodes in the cluster. According to Fig. 2, the result of AI calculation is transmitted to the central server which is flexible in the management of multiple ARM clusters. It is expandable to increase computing power if hardware computing power is insufficient.

Fig. 1. The ARM cluster server.

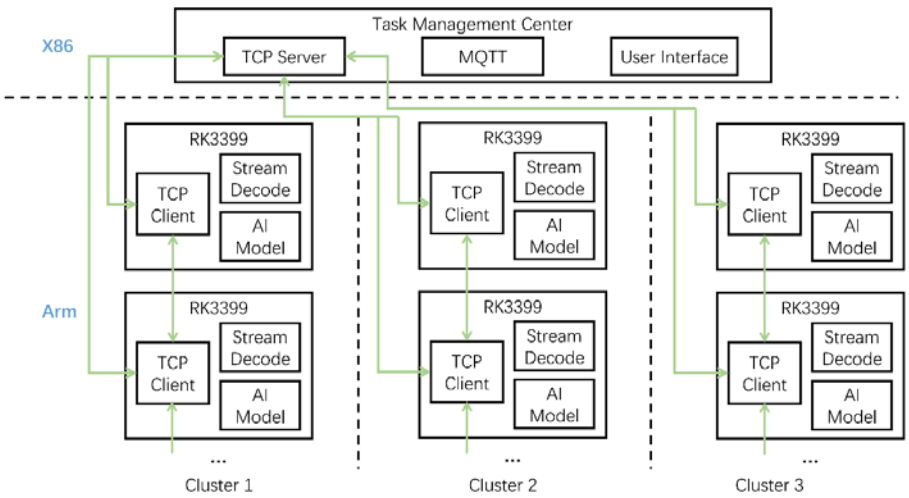

Fig. 2. The architecture of distributed AI computing system.

In this system, the Fork/Join idea is adopted for distributed computing[3]. With multiple arm nodes applied in parallel processing, a real-time AI computing task is split into multiple single-frame image computing tasks. More specifically, prior to being sent to idle nodes in the cluster through the Akka framework, the image frames are encapsulated into TCP commands after online video decoding. Not only is AI computing capable to process video frames of various streams, it is also flexible in adapting AI models to needs. Ultimately, the results of the AI calculation are transferred to the central server through the message queue for the display of data (Fig. 3). On the ARM platform, real-time AI computing tasks can be completed according to this architecture, including such relevant applications as traffic statistics, vehicle classification, and vehicle trajectory tracking.

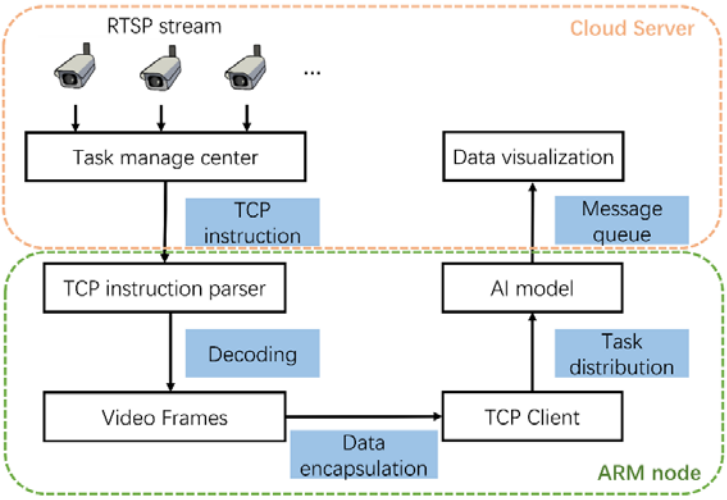

Fig. 3. Workflow of distributed system modules.

3.2 System modules

3.2.1 Task Management Center

Such significant functions as task distribution, node monitoring, dynamic load balancing, and fault tolerance processing are performed by the core module of the whole system. It is responsible for the deployment of code in the cluster, assignment of tasks to each worker node, and keeping track of how tasks are performed. The task management module is purposed to ensure that each processing node is able to receive processing tasks and run at full capacity. As for the task management module, it is necessary for all tasks to be decomposed and dispatched. Besides, all computing nodes in the cluster should be managed in a unified way, and the data should be processed by the business module according to the equalization algorithm. The long link communication is made with each node in each Arm cluster through built-in cluster terminal long link management service. To manage the health and status data of all nodes, the TCP private encryption protocol is applied, so as to accomplish the uplink and downlink of data through two-way communication.

3.2.2 Communication module

Intended for data reception and transmission, it carries out the underlying data transmission. The deployment of each ARM node is carried out using a daemon which is purposed to act as the internal TCP-server and external TCP-client, in addition to maintaining the external and internal data channels of each node. Through decentralization, a service network is formed by the nodes in the cluster in line with the gossip protocol. With its own service capabilities, each node informs each other of its own AI capabilities, service status, uniformly monitors and schedules tasks. In order to describe the information to be communicated, encrypted TCP commands are applied between nodes. The definition of a task is as follows: a specified video stream is pulled and motion is monitored on the video stream, while the video frame is extracted for AI computing in case that a change is detected. The frames to be processed are sent over to complete the AI computing task collectively in case of a backlog of tasks, through the search for other idle nodes with the same AI capability in the network.

3.2.3 Video decoding and stream management

Online RTSP video stream management and codec module, primarily running on the VPU of the ARM node. It is necessary to decapsulate and decode the RTSP video stream, with each frame of image sent to the AI calculation module. With the capabilities of hardware encoding and decoding, the RK3399 is effective in improving performance through the release of CPU resources to the AI module. In the meantime, it is necessary to ensure swift response and reconnection for avoiding packet loss and flash interruption in online video. Thus, the stability and applicability of module are required, with limited resource usage.

3.2.4 Distributed AI module

As the primary module to perform AI computing, it is capable of providing multi-scenario target detection algorithm capability output for application layer services. There are multiple sets of AI models deployed for each ARM node, so as to meet distributed computing with various scenarios and multiple goals. The distributed AI computing module will be flexibly switched to facilitate task scheduling and the management will be unified by the task management center in line with the instructions issued by it.

3.3 Key technologies needed to implement the framework

3.3.1 TCP command parser

A self-defined TCP private encryption instruction set is adopted for the task distribution in this system. The TCP protocol is advantageous due to not only the orderly and reliable data, but also the tiny time delay[4]. The instruction set is a set of languages applied to make description of the communication made between the service layer and the boards in the cluster. For instance, it is necessary to issue a task to the node to obtain the current running status of a service on the board if the service layer wants to know it. The instructions in the instruction set can help express this task. With regard to the instruction set parser, it is involved in the translation of instructions, with each node in the cluster having its own instruction set parser. The instruction set parser performs two major functions. One is to complete the management and monitoring of internal services, for example, service self-discovery, service status monitoring and service management. The other is to synchronize information with the service layer, for example, node registration, sending node status to the service layer, and synchronizing information between the service layer and the node.

3.3.2 Akka cluster based on gossip protocol

As a concurrency framework constructed on the JVM and premised on the Actor model, Akka creates a superior platform for the responsive concurrent applications which are highly scalable and flexible[5]. Featuring a higher abstraction of the concurrency model, the Actor model is an asynchronous, non-blocking, high-performance event-driven programming model. Through the gossip protocol, the boards in the cluster form a service node network in Akka cluster. The decentralized state and load synchronization in the cluster is achieved through the Akka cluster based on the gossip protocol in each cluster node. The AI services running on the board can be well managed through the Akka framework. Besides, for better completion of the cluster-based functional distributed AI computing, the local-group-cluster three-level distributed task allocation system can be applied.

3.3.3 Message queue

In a distributed system, message queuing plays a significant role. The use of message queuing is purposed mainly to enhance system performance, mitigate system coupling, and flatten traffic peaks by means of asynchronous processing[6]. Using EMQ products in this framework, the MQTT Broker is primarily applied to obtain the result output of all AI modules. It is through the MQTT message queue that the output results are transferred to the central server. With the relationship between AI and business system decoupled, high availability and high concurrency are ensured for AI services.

4 System performance analysis

The bandwidth within the Arm cluster is 100Mbps while the bandwidth between the central server and an ARM cluster server is 1000Mbps under this system. At the time of test, the real-time parallel detection of up to 80 video streams is supported by a single cluster. For single-node and cluster tests, a number of AI algorithms with high computing power requirements are selected for comparison against the GPU results. According to Tables 1 and 2, there could be a 20-fold increase in AI computing speed that take tens of seconds to complete on single ARM board with the support of distributed AI clusters, thus producing an effect that matches GPU, which makes it suited to real-time video data analysis.

Table 1. The speed measures end-to-end time per 25 images (opencv4.3.0 + DNN).

Model	Image resolution	RK3399 single board speed	Distributed cluster speed	GPU
YOLOv4-tiny	416x416	15s	0.83s	0.425s
YOLOv4	416x416	63s	4.2s	2.13s

Table 2. The speed measures end-to-end time per 25 images (torch1.8.0 + torchvision0.9.0).

Model	Image resolution	RK3399 single board speed	Distributed cluster speed	GPU
YOLOv5s	640x480	47.3s	2.36s	0.33s
YOLOv5m	640x480	128s	5.3s	0.62s
YOLOv5l	640x480	215s	8.72s	0.76s
YOLOv5x	640x480	317s	13s	0.96s

5 Conclusion

As the artificial intelligence industry advances rapidly, AI has been applied widely to speed up industry changes, and the demand for AI computing on the edge is increasingly diversified. Herein, a real-time video analysis system at the edge based on a distributed AI cluster is proposed. Following numerous tests and verifications, it is demonstrated that the system is advantageous from multiple angles: (1) The system is capable to be adaptively expanded and deployed at the edge; (2) This system is effective in reducing the requirements on bandwidth, and in enhancing data security; (3) Distributed AI computing contributes to improving computing efficiency and completing the heavy-duty real-time AI tasks assigned at the edge.

This work was supported by National Key R&D Project of China (No.2019YFB1600700).

References

1. Markit, I. "The top trends of 2019: powered by transformative technologies." *IHS Markit* (2019).
2. Xue, Cheng-Xin, et al. "A 22nm 4Mb 8b-Precision ReRAM Computing-in-Memory Macro with 11.91 to 195.7 TOPS/W for Tiny AI Edge Devices." *2021 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 64. IEEE, 2021.
3. Garibay-Martínez, Ricardo, et al. "Improved holistic analysis for fork–join distributed real-time tasks supported by the FTT-SE protocol." *IEEE Transactions on Industrial Informatics* 12.5 (2016): 1865-1876.
4. Forouzan, Behrouz A. *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002.
5. Srirama, Satish Narayana, Freddy Marcelo Surriabre Dick, and Mainak Adhikari. "Akka framework based on the Actor model for executing distributed Fog Computing applications." *Future Generation Computer Systems* 117 (2021): 439-452.
6. Light, Roger A. "Mosquitto: server and client implementation of the MQTT protocol." *Journal of Open Source Software* 2.13 (2017): 265.