An empirical study on big video data processing: architectural styles, issues, and challenges

Weishan Zhang, Zhichao Wang, Liang Xu, Dehai Zhao, Faming Gong, and Qinghua Lu
School of Computer and Communication Engineering, China University of Petroleum
No.66 Changjiang West Road, Qingdao, China

Abstract

Video data contributes to the majority of big data, henceforth, how to efficiently and effectively discovering knowledge from large-scale video data becomes a crucial challenge. In this paper, we propose multiple architectural styles for the domain of large-scale video data analytics services. These styles include online combined with offline processing style, distributed shared repositories, image mining and prediction services with deep learning techniques. These architectural styles are successfully implemented and examined in a number of domains including smart traffic and smart drones, as demonstrated in a middleware developed specifically for large-scale continuous video data processing.

Keywords

Cloud Computing, big video data processing, architecture styles, Smart City.

I. Introduction

Video data grow rapidly and become the majority of big data. How to process these big video data efficiently and effectively to extract useful knowledge is challenging, due to the volume, velocity and other complexities of video data. From the point view of software architecture [1], there naturally arise the question on how to build an effective architecture for big video data processing.

In literature, there has been a lot of work on software architecture as surveyed recently [2], which has listed MapReduce as the key architectural style for big data processing. The architectural design for big data systems is now attracting research efforts, e.g. a reference architecture for big data systems is proposed in [3] as a general architecture for all domains. The problem of such a reference architecture is that a reference implementation is needed to make it really useful for a specific domain. Therefore, work on domain specific big data architecture arises, e.g. SOLID architecture for real-time management of big semantic data as in [4], and also the Lamda architecture as proposed in [5]. However, these domain specific architectures do not consider low level architectural styles of that domain, which are essential parts for achieving high quality software, not even architectural styles suitable for big video data domain.

To address these issues, this paper proposes four architectural styles for the big video data processing domain, with the intention to improve the quality of software architecture design of big video data mining and processing. Due to the success of deep learning [6] techniques for image and video processing recently [7] [8], we rely heavily on deep learning for video mining and prediction of cloud resources request during the design of the proposed architectural styles.

These architectural styles include OCOP style (online combined with offline processing style), DSR style (distributed shared repository style), DLIM style (deep learning based image mining style), CMA style (Cloud metrics analytics style). They are successfully implemented and utilized in a number of domains including smart traffic and smart drones, as demonstrated in a middleware developed specifically for big video data processing.

II. Architecture Styles for big video data processing

We propose four architectural styles to address the storage and sharing of video data, the processing of video data including historical and current data, the mining of video data, and also the analysis of cluster running status using deep learning and aspect oriented programming.

A. OCOP style (Online combined with offline processing style)

1) Scenario: Big video data processing may involve the processing of large amount of historical video data, and also the processing of most current data. Knowledge on both historical data and current data may be needed, but can be mined separately. The structure of this style is shown in Figure 1.

2) Design elements:

- Components: Video data component, Video storage component, Video offline processing component, Video real-time processing component, Service component.
- Connectors: procedure calls, Distribution, Streaming, Shared memory.
- Data: Video streaming, arguments passed to methods, training parameters.

3) Topology and constraints: The topology can be Linear and/or Layered. Real time processing and offline processing components can be used separately. It can be dynamically extended where video data component can register devices and delete devices dynamically.

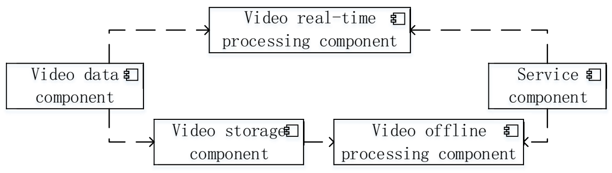
Video real-time processing component
Video data component
Service component
Video storage component
Video offline processing component

Figure 1. Online combined with offline processing style

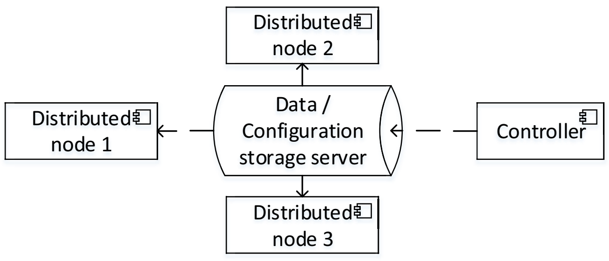

Figure 2. Distributed shared repository style

4) Examples of usage: Handling a large number of surveillance cameras: smart traffic, smart drone, and smart home.

5) Advantages and disadvantages: Each component is independent of each other and it has clear video processing steps. At the same time this style makes sure that good performance of video processing can be assured. On the other hand, the whole structure is complex.

B. DSR style (Distributed shared repository style)

1) Scenario: Distributed video processing needs different kinds of processing components. These components will share a part of data (original video information, the network and weight information of deep learning). Controller is responsible for produce shared data, shared data server is used for storing shared data and ClientNode consumes these data. The structure of this style is shown in Figure 2.

2) Design elements:

- Components: controller, shared data server, ClientNode.
- Connectors: procedure calls, Distribution, Shared memory, database queries.
- Data: arguments passed to methods, training parameters.

3) Topology and constraints: The topology of this style can be in a Star manner, where multiple clients can making requests to the server and get data.

4) Examples of usage: A large number of video data mining: video website, video analysis of social networking sites.

5) Advantages and disadvantages: Data can be shared and reused. However, interaction between independent programs need a complex regulation. And also data on the distributed shared repository subject to frequent changes (and requires propagation to all other components).

C. DLIM style (Deep learning based image mining style)

1) Scenario: Users only need to input video data and then get the mining results, without considering details of internal processing. The internal processing can be realized with some tools, using which users can design and construct different deep neural networks. Each neural network has a configuration to control its running status. New training data is used to update network models based on the current model so that the network model can fit new data. When input images which are raw data from media server or cameras pass through the package, it can get mining results. The structure of this style is shown in Figure 3.

2) Design elements:

- Components: Development tools, deep neural networks and configuration.
- Connector: Data flow, shared data.
- Data: Training data contains images and their labels. Input images can be single picture or continuous frames and the output results are refined information mined from image data.

3) Topology and constraints: All neural networks are different in the running package. All connections inside the package are unidirectional.

4) Examples of usage: Image recognition using deep neural networks, and object detection using deep neural networks.

5) Advantages and disadvantages: It is unnecessary to design complex connections because the architecture is simple and explicit. It can be used for image processing and data mining. However, training deep neural networks and adjusting parameters cost a lot of time and energy, and massive manual work is unavoidable.

D. CMA style (Cloud metrics analytics style)

1) Scenario: The scenario is stimulated when metrics of a task and also the status of a cloud cluster are sampled. Those metrics are periodically transferred to the monitor through a remote procedure call. These collected metrics are analyzed and generate index to the health condition of the task. These metrics are analyzed through a deep learning based prediction, and then an Actuator is used to conduct corresponding actions based on the prediction results. The structure of this style is shown in Figure 4.

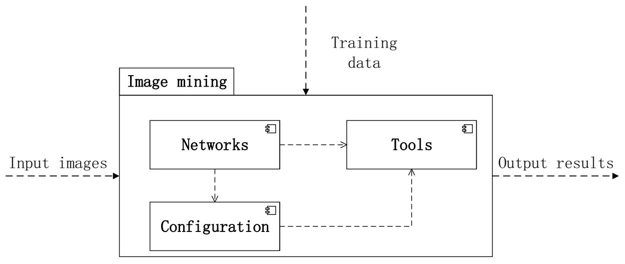

Figure 3. Deep learning based image mining style

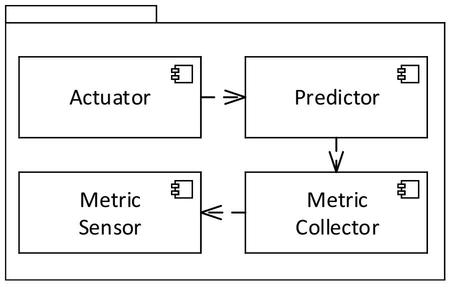

Figure 4. Cloud metrics analytics style

2) Design elements:

- Components:Metric Sensor, Metric Collector, Predictor, Actuator.
- Connector: RPC (Remote Procedure Call), AOP (Aspect-oriented Programming).
- Data: Metrics of tasks running on Big Data environment.

3) Topology and constraints: Each task will be devised a metric sensor which periodically samples metrics from the task through AOP protocol. The obtained metrics are for real-time prediction which is done in the predictor. If some abnormal patterns are found in metrics, a native method call will be conducted to do higher-level decision makings which is the responsibility of the actuator.

4) Examples of usage: Resource request prediction, scheduling between heterogeneous processors.

5) Advantages and disadvantages: It modularizes the functionality of Cloud metrics analytics. Considering the computing performance, resource-intensive algorithms for metric analytics are not allowed to be in the proposed architectural style.

III. A SMART BIG VIDEO DATA PROCESSING MIDDLEWARE USING THE PROPOSED ARCHITECTURAL STYLES

Considering the proposed architectural styles, and the complexities of big video data, we design a middleware for smart big video data processing. We choose open source software packages to implement the middleware in order to make a reference implementation that accessible by researchers. The architecture of the middleware as shown in Figure 5.

This architecture is in general a layered architecture. The bottom layer is a generalization of video data collection, where a video streaming server and a camera API component are used to receive different type of video data from different cameras.

The second layer is called Data Processing layer, which is actually composed by two sub-packages, namely the package of Offline Processing based on Apache Hadoop, and the other package of Online Processing based on Apache Storm stream processing. Considering the nature of different video processing tasks, some features like video summary, video encoding/decoding for large amount of video data are within the scope of offline processing where no real-time requirements are needed. Some important deep learning offline training components are located in this package, which include DCNN training, DBN (Deep Belief Network) training. Image mining components outputs are stored into a storage serverfor data mining by offline and real-time layer. Some lightweight video processing tasks are assigned to the Online Processing package, for example background subtraction. In this package, the recognition at real-time is achieved based on the training results from the offline neural network training, including the recognition of events, objects and behaviors. Additionally, there are some services regarding the obtaining of the middleware running status, especially the detection of bottlenecks, and then rescheduling resources when there are problems.

Built on the processing results from Data Processing layer, there is a Data Service layer, which is used to abstract the underlying processing capabilities.

IV. EVALUATION

The evaluations are conducted in different domains, and we choose smart transportation, and smart drone for object detection as evaluating cases. The four architectural styles made contributions to the video data processing middleware in different areas, and we will evaluate our system in performance, scalability and availability.

There are 9 IBM3650 M4 servers deployed as a small scale cloud infrastructure, where each node is running Ubuntu 14.04 server, and a modified version of deep learning tool Caffe is deployed on every node.

A. Performance

In order to validate the performance, a use case of traffic statistics application (TSA) is used as an example. It is used to count the number of cars, motors and pedestrians that pass across one camera in a certain time interval.

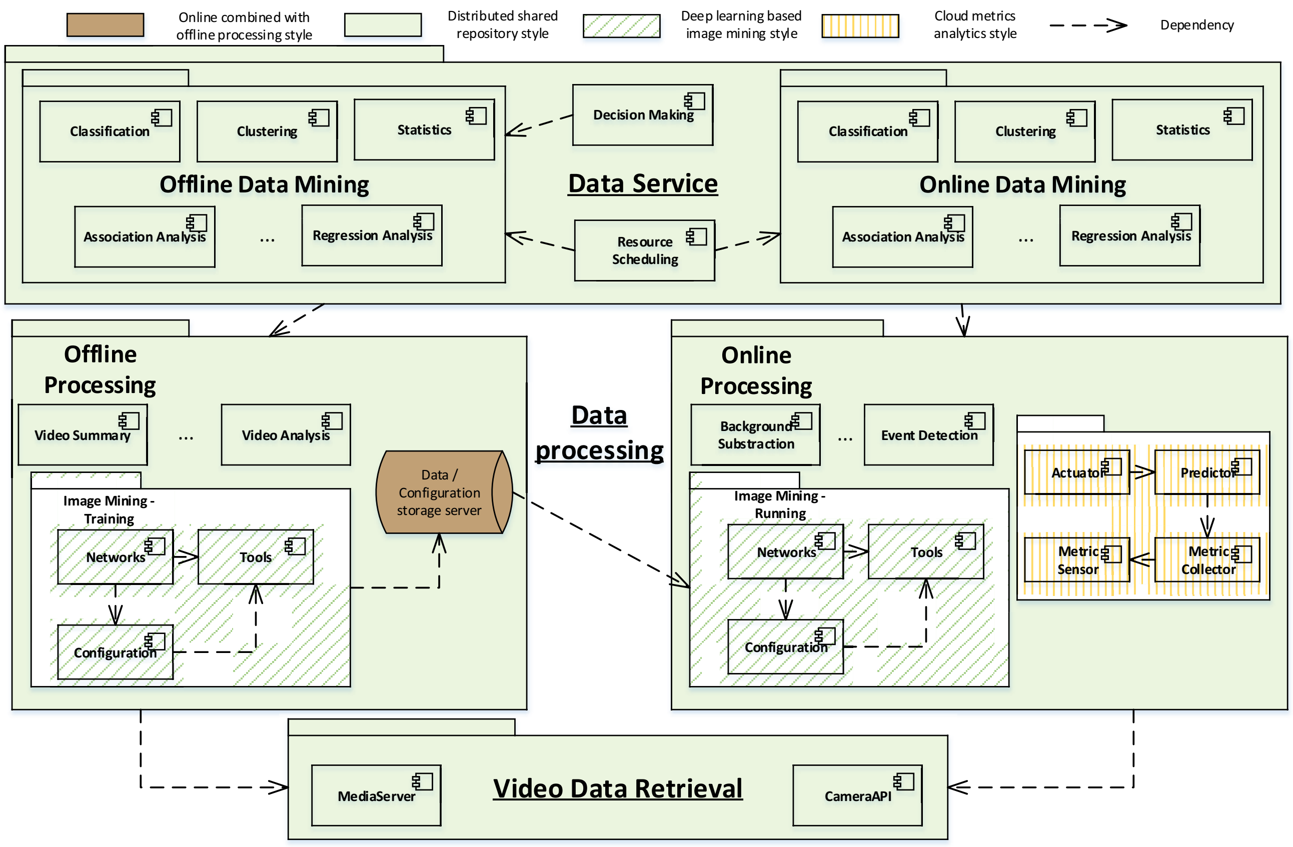

Figure 5. The architecture of smart big video data processing middleware

Table I
PERFORMANCE OF DCNN

No	1	2	3	Average
time	22.655	23.295	22.608	22.853

Table II
PERFORMANCE OF VIDEO SUMMARY

	frame size	video size	processing time
video1	640*480	3072M	1h0m53s
video2	640*480	9216M	1h51m3s
video3	640*480	15360M	2h32m15s

Deep convolution neural network (DCNN) is used in the DLIM style, for object classification in this evaluation case. First, we create a dataset which includes cars, motors and persons on the offline processing layer. There are 46906 training images and 8274 validation images in total. After a number of tests, we choose a DCNN with image size of 32*32, and 5 layers for classification. The well trained neural network is used for real-time classification in the online processing package running on Storm. Using this neural network and Hadoops MapReduce programming model recognize the data of offline layer. In the online processing, we uses the Gaussian mixture model to remove background, then locate the moving target and classify moving targets. Every 10 frames get a set of data, and calculate the processing time of each frame. The performance results are shown in Table I. The performance of aggregating the offline processing and online processing is accomplished by using Splout SQL to query the MapReduce results in Hadoop, and using Trident DRPC to query on the processing results from Storm. The average of corresponding performance results is less than 50ms.

B. Scalability

To test and verify the scalability, in the offline processing layer, we conduct video summary with MapReduce, as shown in Table II. We can see that when the video size increase significantly, the processing time is kept within reasonable time.

We have also tested the scalability of the real-time processing layer to check how many cameras can be processed simultaneously in the middleware. We still use Storm's program in section 4.1 to test scalability, The results are shown in Figure 6, we can see that the performance is good when more cameras are added until the number reach 100, where the program takes around 40 ms which is not very acceptable for real-time requirements.

Figure 6. Scalability with more cameras

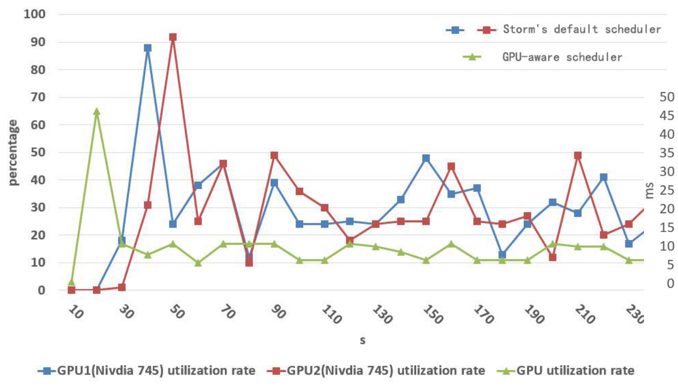

Figure 7. GPU utilization rate by using Storm's default scheduler and GPU-aware scheduler

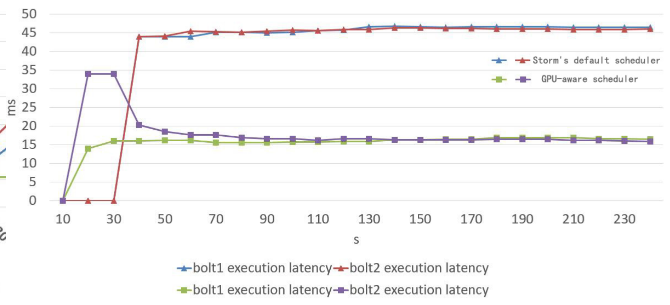

Figure 8. Bolts execution latency by using Storm's default scheduler and GPU-aware scheduler

C. Availability

To ensure availability of the online processing, we run an object recognition topology which will cost a lot GPU. GPU-aware scheduler is serving as the actuator in the CMA style, and deep belief network is used for performing resource request prediction. We use Storms default scheduler as a comparison with our own scheduler. We get the GPU utilization and the bolt execution latency every 10 seconds.

From Figure 7, we can see that, Storm's default scheduler will assign two bolts to two supervisors equipped with GPU745, while GPU-aware scheduler assigns two bolts to a supervisor with a strong GPU. And from Figure 8, we can see that our scheduler can assure much lower executor latency. When we add more bolts, Storm's default scheduler will incur an out of memory error. But the topology runs normally using the GPU-aware scheduler, because when the monitoring system found the bolt execution latency is too high, it then re-balances the topology, and both bolts are assigned to one supervisor with Nvidia 970. After re-balancing, the bolt execution latency is much lower, and it can use GPU resources efficiently. We can conclude that the middleware has a good availability of fault-tolerance.

We had also evaluated the middleware for its availability from the point of cluster nodes failures, and the middleware can elegantly handle such situations where it can still function properly with acceptable performance side effects [9][10].

V. Issues and Challenges

Although these proposed architectural styles can deal with big data processing issues, there are some challenges to be resolved. First of all, network bandwidth is always the key bottleneck when solving large amount of video data in a cluster. If the limitation of network bandwidth can be removed by some specific distribution strategies, there is no doubt that the processing speed will increase in a large degree. Secondly, selecting a suitable processing algorithm according to both functional requirement and cluster performance is also an important point need to be considered. In addition, DLIM style provides an easy way to conduct image mining using deep learning but the network structure and parameters have to be adjusted artificially, which still need a lot of manual work.

CMA gives an architectural style solution for a big video data environment to the perception of the running conditions based on which higher-level decisions can be made such as scheduling and prediction. Since analytics for big video data are data-intensive and computation-intensive, implementing CMA must consider constraints that any artifact in CMA should be lightweight in order not to influence the performance of tasks running.

VI. Related work

Software architecture [2] is very important for efficient big data processing due to characteristics of big data. The proposed a reference architecture for big data systems as in [3] is suitable for all domains. consequently this reference architecture need to be specialized for big video data processing. We have different objective in our paper in that we want to define resuable elements for big video processing in a finer granularity instead of a big bunch of architecture.

Henceforth, we propose different architectural styles to handle different usage scenarios, including data processing, video object recognition, prediction, and so on.

Another close work to what we have done is presented in a solid architecture for real-time management of big semantic data [4]. SOLID consists of an online layer, merge layer, data layer, index layer,and service layer. The main drawback is that SOLID does not separate stream and batch processing, which is important to meet different quality requirements, esp. for different video processing scenarios that can have different performance expectations.

The Lamda architecture as proposed in [5] is motivating us to propose the online combined with offline processing style. Lamda architecture focuses on real time big data processing, which is also part of our goal. However, does not consider low level architectural styles which are essential part for achieving high quality software, and henceforth no architectural styles suitable for big video data domain are there.

In our previous work [10], we have addressed the issue of how to make use of batch processing and stream processing to implement online combined with offline processing style, motivated by Lamda architecture [5]. From our experiences on big video processing, we found that there arises needs for intelligent processing. This motivates us to propose more architectural styles for video data mining and prediction, which are very important for big video data processing.

VII. Conclusion and future work

Video data become the majority of big data. Designing an efficient architecture for video processing is challenging due to volume, velocity, and other complexities like intrinsic video complexities. In this paper, we propose four architectural styles to address the storage and sharing of video data, the processing of video data including historical and current data, the mining of video data, and also the analysis of cluster running status using deep learning and aspect oriented programming. These architectural styles are combined and implemented in a big data processing middleware, and are evaluated in terms of performance, scalability, modifiability, and availability. The evaluations are conducted at different domains to show the effectiveness of the proposed architectural styles, namely smart transportation, and smart drone.

Domain specific software reuse is one of the most successful direction for software engineering. We consider that this is also true for big data processing. We are investigating the abstraction of more domain specific architectural styles for big data processing, as a lot of these styles are very related to the domains they are addressing.

Acknowledgments

This research was supported by the Program on Innovative Methods of Work from Ministry of Science and Technology, China (Grant No. 2015010300).

References

[1] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software architecture: foundations, theory, and practice*. Wiley Publishing, 2009.

[2] A. Sharma, M. Kumar, and S. Agarwal, “A complete survey on software architectural styles and patterns,” *Procedia Computer Science*, vol. 70, pp. 16–28, 2015.

[3] P. Pääkkönen and D. Pakkala, “Reference architecture and classification of technologies, products and services for big data systems,” *Big Data Research*, vol. 2, no. 4, pp. 166–186, 2015.

[4] M. A. Martínez-Prieto, C. E. Cuesta, M. Arias, and J. D. Fernández, “The solid architecture for real-time management of big semantic data,” *Future Generation Computer Systems*, vol. 47, pp. 62–79, 2015.

[5] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[6] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[7] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[8] W. Zhang, P. Duan, Z. Li, Q. Lu, W. Gong, and S. Yang, “A deep awareness framework for pervasive video cloud,” *IEEE Access*, vol. 3, pp. 2227–2237, 2015. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2015.2497278

[9] W. Zhang, L. Xu, Z. Li, Q. Lu, and Y. Liu, “A deep-intelligence framework for online video processing,” *IEEE Software*, vol. 33, no. 2, pp. 44–51, 2016. [Online]. Available: http://dx.doi.org/10.1109/MS.2016.31

[10] W. Zhang, L. Xu, P. Duan, W. Gong, Q. Lu, and S. Yang, “A video cloud platform combing online and offline cloud computing technologies,” *Personal and Ubiquitous Computing*, vol. 19, no. 7, pp. 1099–1110, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00779-015-0879-3