Poster: Context Aware Dynamic Log Chunking for Mobile Healthcare Applications

Rahul Krishnan
Amrita Center for Wireless Networks & Applications (AmritaWNA)

Bithin Alangot
Amrita Center for Cybersecurity Systems & Networks

Venkat Rangan
Amrita Center for Wireless Networks & Applications (AmritaWNA)

Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham, Amrita University, India
{rahulkrishnan, bithina}@am.amrita.edu, venkat@amrita.edu

ABSTRACT

Time series data from sensor devices are increasingly stored in log data structures across the cloud and mobile devices. Currently, log data is accessed as chunks of fixed size, which enhances performance by prefetching of data. However, in applications such as remote monitoring of patients using mobile devices, data requirement of end users varies significantly depending upon their roles. The fixed chunking approach would lead to unnecessary data download due to the dynamic variability of data access. Also, the requests are more often than not based on fixed time chunks that do not necessarily translate to fixed data size. To overcome this challenge, we present a dynamic log chunking mechanism based on reader access pattern and domain specific data characteristics. The application of this method in the area of remote patient monitoring in bandwidth starved rural areas is shown to result in bandwidth and cost savings of 14% without affecting the prefetch performance.

CCS Concepts

•**Information systems** → *Distributed storage;* •**Networks** → *Network performance analysis;*

Keywords

Log storage; Edge computing; Data retrieval

1. INTRODUCTION

Log data abstraction is fast becoming the most efficient way to manage time-series sensor data in an IoT-Cloud environment [4]. Logs allow data access in fixed size chunks which introduces efficiency since the downloaded chunk includes prefetch data that will most likely be used in subsequent user requests [3]. The fixed size chunking is also preferred due to its simplicity in implementation and data management across a wide range of applications. However, in many applications such as remote patient vitals monitoring, data reads occur in different size granularity depending upon the end users; such as doctors, clinicians, nurses, technicians, and patients. It was seen from experiments that a global or even application wide constant chunk size introduces inefficiencies which far outweigh its benefits. The difference is pronounced specifically in bandwidth starved environments with barely minimum cellular data coverage such as in rural regions. Hence, we put forth a dynamic log chunking mechanism based on reader's data access pattern. The evaluation was conducted using data from a remote ECG monitoring device [2], and showed significant improvement over fixed chunking.

MobiHoc'16 July 04-08, 2016, Paderborn, Germany
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4184-4/16/07.
DOI: http://dx.doi.org/10.1145/2942358.2942406

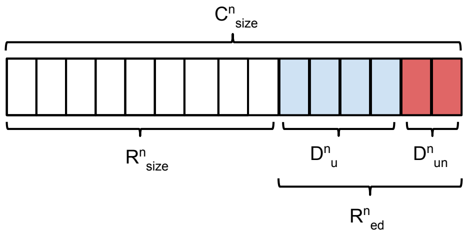

Figure 1: Representation of a log chunk.

2. DYNAMIC LOG CHUNKING

The log readers request, R_{size} of data from the remote nodes (cloud and edge device). The application downloads this data in chunks of size, C_{size} (refer Fig. 1). The downloaded chunk has two parts: the requested data of size R_{size} and extra prefetch data R_{ed}. The extra data can be classified into two: useful data of size D_u which will be used by the user in future requests, and the rest of the data of size D_{un} which is the prefetched data that would not be used by the reader in subsequent requests. The relationship between D_{un} and D_u is a direct measure of the reader's access pattern, and hence we change the chunk size for the $(n+1)^{th}$ request, C^{n+1}_{size}, based on the current chunk size C^n_{size}. We formulate it as:

$$\boldsymbol{C^{n+1}_{size} = C^n_{size} - D^n_{un}},\ \ if\ D^n_{un} > 0\ . \tag{1}$$

$$\boldsymbol{C_{size}^{n+1}} = \boldsymbol{C_{size}^{n}} + \boldsymbol{\alpha} * \boldsymbol{D_u^n}, \;\; if\ D_{un}^n = 0\ and\ D_u^n > 0\ . \quad (2)$$

$$\boldsymbol{C_{size}^{n+1}} = \boldsymbol{C_{size}^{n}} + \boldsymbol{\beta} * \boldsymbol{R_{size}^{n}}, \;\; if\ D_{un}^n = 0\ and\ D_u^n = 0\ . \quad (3)$$

Equation (1) is used when D_{un} is greater than zero and it effectively reduces the unused data from the succeeding chunk size. Equation (2) is applied when there is no unused data in the current request, which implies that all the prefetch data was useful. Hence, we increase the prefetch size by using a growth rate α. Equation (3) is used when both the used and the unused data in the chunk is zero. From this we infer that the request size was equal to the chunk size, thereby leaving no space for any prefetch data. Hence, we can increase the chunk size by a factor β of the current request size. We have evaluated two models for updation of the new chunk size:

- **Dynamic Chunking 1 (DC 1)** The equations (1) to (3) are applied based on the average of D_{un}, D_u and R_{size} over n requests. Using these values, a new chunk size C_{size} is calculated and used for the next n requests.

- **Dynamic Chunking 2 (DC 2)** C_{size} is calculated after each request but not updated until a predefined number of requests, n, are completed. The average of C_{size} over n requests is calculated and then used as the new chunk size for the next n requests.

3. EVALUATION AND RESULTS

We compared D_{un} in dynamic chunking and fixed size chunking models. The latter model used 384 KB (two minutes of 3-lead ECG sensor data) as chunk size. The request sizes were modelled according to the commonly accessed pattern of ECG data by the doctors in a tertiary hospital. We observed that only two minutes of ECG data was being analyzed by them at a given point of time. Hence, R_{size} was picked up from normally distributed data request sizes, with mean as 230 KB and standard deviation of 40 KB. The used prefetch data D_u also varied with a mean of 60% of the extra data R_{ed} and standard deviation of 10%. A total of 100 requests, adding up to 35 MB of data (four hours of 3-lead ECG sensor data) were considered for the evaluation. The values for other parameters were: α=0.3 and β=0.1. These values can be varied based on application, data and role specific parameters. In effect it controls the growth rate and has direct effect on the time taken for convergence towards an optimum log size. Our choice of these values gave optimum savings in ECG sensor data visualization application for the given user role and data characteristics. Figure 2 shows R_{ed}, D_u and D_{un} as a percentage of the chunk size while using the fixed chunking method and two approaches of the dynamic chunking model. In both models the chunk size was updated after n=10 requests. We noticed that the reduction of unused prefetch data implies overall reduction in download and upload requirements at the cloud as well as the edge nodes. Table 1 lists the data downloaded from 100 requests. The bandwidth savings from DC1 and DC2 are 20% and 14% respectively. These data saving translates to similar cost savings too. For instance, Amazon S3 charges \$0.09/GB for outbound data bandwidth for up to 1TB per month. A reduction of 14% in out-bound traffic would result in savings of \$126 per month. On the other hand, for the end user who uses mobile data network for accessing patient

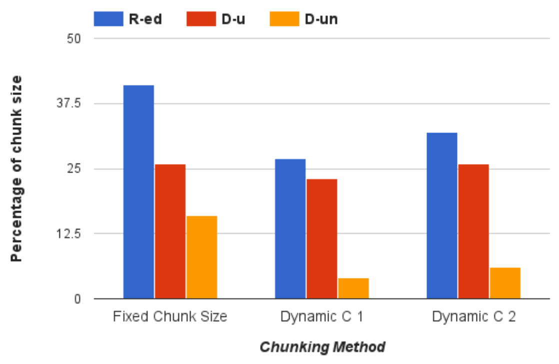

Figure 2: Comparison of dynamic chunking and fixed chunking models.

Table 1: Data usage comparison (in KB) using dynamic chunking (DC 1 & DC 2) and fixed chunking (FC)

	R_{size}	R_{ed}	Total Data	Savings %
FC	22,546	15,854	38,400	-
DC 1	22,546	8,304	30,850	19.7
DC 2	22,546	10,584	33,130	13.7

data, a savings of 14% translates to around \$10/GB; a significant saving considering patients in developing countries spend a large portion of their income for data access [1].

4. CONCLUSION AND FUTURE WORK

Dynamic chunking is shown to reduce bandwidth usage and cost incurred in mobile cloud infrastructure especially for applications where reader's access pattern varies. Furthermore, in the final poster, we intend to present a complete evaluation of an application wide semi-static chunk size in comparison with the dynamic chunking technique.

5. REFERENCES

[1] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin. Flywheel: Google's data compression proxy for the mobile web. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, pages 367–380. USENIX Association, 2015.

[2] N. Dilraj, K. Rakesh, K. Rahul, and M. Ramesh. A low cost remote cardiac monitoring framework for rural regions. In *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*, pages 231–236. ICST, 2015.

[3] T. Gupta, R. P. Singh, A. Phanishayee, J. Jung, and R. Mahajan. Bolt: data management for connected homes. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 243–256. USENIX Association, 2014.

[4] I. Shafer, R. R. Sambasivan, A. Rowe, and G. R. Ganger. Specialized storage for big numeric time series. In *Proceedings of the 5th USENIX conference on Hot Topics in Storage and File Systems*, pages 15–15. USENIX Association, 2013.