Optimization of request processing times for a heterogeneous data aggregation platform

Victoria Tokareva
Karlsruhe Institute of Technology, Institute for Astroparticle Physics, 76021 Karlsruhe, Germany
E-mail: victoria.tokareva@kit.edu

Abstract. A heterogeneous data aggregation system, e.g. developed within the frame of the GRADLC project, allows for a flexible expansion by connecting new data storages, as well as providing researchers a fast and aggregated access to heterogeneous data from independent (astroparticle physics) projects, while reducing the load on the original data storages. However, this flexibility requires balancing user requests in the queue with respect to various request processing times for the distributed storages, taking into account the different data processing policies on each particular storage. In order to attack this problem, a mathematical model of the data aggregation system was developed, and approaches to optimization of the request ordering in the processing queue are proposed and investigated by performing a numerical experiment. Based on this results, a job shop scheduling algorithm was revealed which gives benefit in mean request processing times compared to the well-known first in, first out (FIFO) model.

1. Introduction
In the last decade, trends in open data access to be used for interdisciplinary research have become relevant in science all over the world [1, 2, 3, 4]. The globalization of science leads to the exchange of experience and ideas between different fields of knowledge, and allows us to expand the horizons of our understanding of processes of different origins by extending the range of methods and techniques for obtaining scientific knowledge. At the same time, researchers of interdisciplinary subjects continue to face problems such as an unsystematic approach to store and provide data, or the lack of a single interface for data access [2, 5, 6, 7].

Currently existing approaches for working with distributed data storages [8, 9, 10] imply that the remote storage use a single file system (for example, HDFS or CVMFS) as well as a unified
software stack for data processing, which require significant costs and changes to the streamlined workflow of remote data centers.

In the German-Russian Astroparticle Data Life Cycle (GRADLC) [6] project we are developing an alternative approach: a system that treats remote storage servers as black boxes, an aggregator carries the load of handling integrated user queries (see the general scheme in Fig. 1). Similarly to the aforementioned solutions [8, 9] our system uses the metadata to optimize the processing time of requests.

To further improve the quality of the request service, a mathematical modeling of the system behavior turned out to be relevant in order to identify the most productive heuristics to reduce the waiting time in the queue. In this paper, we will consider a static model for processing user requests in a system with an aggregator and several storages in the case of a simple user behavior scenario, described in chapter 2.

2. Description of the system operation and the mathematical model
Let users of the system send requests $R_j(s, p)$ to the aggregator, where $j \in [0, ..., J]$ is the number of the request in the system, $s_i, i \in [1, ..., S]$ are the remote storage identifier, and $p \in \mathbb{R}^n$ are the data selection parameters of the request. Further on, the user application goes through the fundamental stages of data processing shown in Fig. 2.

We will consider the case when one user is interested in the data of only one storage, i.e. when $s$ is a scalar.

For such a case, one can create $S$ parallel queues for execution and process $R_j(s, p)$ requests for different storages $S$ in parallel, as shown in detail in Fig. 3.

It should be noted that this case is not representative for the main class of problems solved by the aggregator, since the purpose of creating such systems is to provide users with joint information from different sources. However, developing a model for a given simple case allows us to look at the relationships between objects in the system and highlight the essential patterns of the processes that occur, which can later be supplemented and expanded to account for more realistic use cases.

We will rely on the fact that the criteria $p$ of the user request $R_j(s, p)$ are encapsulated in it and are not explicitly known to the scheduler. The execution times of the processing steps of the request shown in Fig. 3 are not immediately known to the scheduler. They depend linearly on the

![Image](https://via.placeholder.com/150)
number of entries $n_{ij}$ in the remote storage $s$ that correspond to the request parameters $p$, and the scheduler may need to perform time-consuming computations to estimate these numbers.

Let us consider as an example the individual stages of request processing by different storages, using some of the remote storages connected to the GRADLC system (Fig. 4).

Thus, we can conclude that functions (1) can be defined for the considered system of aggregated information collection from distributed storages.

$$
\begin{align*}
    t_s(n_{ij}) &= \nu \cdot n_{ij}, \quad \nu \in \mathbb{R} \\
    t_f(n_{ij}) &= \mu^s \cdot n_{ij}, \quad \mu = (\mu_1, \ldots, \mu_s) \in \mathbb{R}^s \\
    t_a(n_{ij}) &= \tau \cdot n_{ij}, \quad \tau \in \mathbb{R}
\end{align*}
$$

Let us introduce the procedure $C$, which maps vectors from the space of parameters $p$ to $n_{ij} \in \mathbb{R}$. This procedure will allow performing a preliminary calculation of the request execution time $R_j(s, p)$, which is necessary for further optimization of the queue ordering. The execution time $t_c$ of the procedure $C$ is non-uniformly distributed in $[0, T_c]$. When planning the request execution order, the call to the $C(p)$ procedure creates overhead costs that must be compensated for by optimizing the queue waiting time of requests, which may not be fulfilled in the case of a small number of requests to the system.

**Figure 3.** Principal stages of processing the request $R_j(s, p)$ by the aggregator.

**Figure 4.** Dependencies of the execution times on the number of requested records $n_{ij}$ for various request processing stages: a) a dependence of the time $t_s(n_{ij})$ of query processing by the metadata database on the number of requested records $n_{ij}$, approximated by a linear equation on a logarithmic scale; b) a dependence of the time $t_f(n_{ij})$ of retrieving the requested records on their number $n_{ij}$, approximated by a linear equation in a logarithmic scale; c) a dependence of the time $t_a(n_{ij})$ of post-processing of a query on the aggregator side on the number of requested records $n_{ij}$, approximated by a linear equation in a logarithmic scale.
Based on the dependencies described above and the request processing model shown in Fig. 3, we obtain the following factors for calculating the expected waiting time of a request in the system:

0) Performing an estimation $C(p) = n_{ij}, t_c \in \text{unif}(0, T_c)$
1) Waiting time of a request in the queue $t_{ij}^q$
2) Initializing a query to the metadata database $t_{in} \in \text{unif}(0, \Theta_{in})$
3) Processing of a request by the metadata database $t_{ij}^s = \nu \cdot n_{ij}, \nu \in \mathbb{R}$
4) Time of retrieving requested records from a remote storage $t_{ij}^f(n_{ij}) = \mu_i \cdot n_{ij}$, $\mu = (\mu_1, ..., \mu_s) \in \mathbb{R}^s$
5) Request post-processing time on the aggregator side $t_{ij}^a(n_{ij}) = \tau \cdot n_{ij}, \tau \in \mathbb{R}$

Here steps 3–5 can be performed in parallel in blocks as a part of a single request, while steps 0–3 must be performed sequentially. From Fig. 4 we see that the UUID fetching time for a single event is about an order of magnitude less than the one event fetching and post-processing. The competition, that is most expensive in terms of time and system resources, occurs when processing multiple requests to the same remote storage.

Thus, the execution time of the request $R_{ij}$:

$$T_{ij} = t_c + t_{ij}^q + t_{in} + n_{ij} \cdot \max(\nu, \mu_i, \tau)$$ (2)

From (2) we have:

$$t_{ij}^q = \sum_{d=1}^{j-1} T_{ij}, j = 2, J, \quad T_{i1} \in \mathbb{R}$$ (3)

We are looking for a request application processing schedule such that:

$$\sum_{j=1}^{J} t_{ij}^q \rightarrow \min, \quad i \in [0, S]$$ (4)

3. Simulations

Known strategies [11] for distributing jobs in a queue are:

(i) FIFO (first in, first out).
(ii) LIFO (last in, first out).
(iii) Capacity: the approach is based on the establishment of sub-queues for “large” and “small” tasks. It assumes the concurrent access of several requests to the storage, which leads to overhead costs and slows down the search and retrieval of data in our case.
(iv) Fairness: the model assuming parallel processing of several requests and processing requests with interruptions, which is unacceptable within the framework of the system under consideration.
(v) Priority: a priority of the job is evaluated based on known parameters and the jobs are ranked based on the assigned priority. There are various ways to assess the priority of a task. In this paper, ranking by task execution time was considered.

Within this study, we performed a simulation of the modeling, the results are shown in Fig. 5. The figure shows that with a relatively small number of tasks, the overhead of calculating the expected duration of the task and the redistribution of tasks in the queue affect the performance of the ranking algorithm in such a way that it does not show a significant advantage. However,
as the number of tasks grows, the ranking algorithm begins to outperform the FIFO approach confidently. Qualitatively this happens because the first jobs contribute more to the total waiting time of all requests than the last ones, so it is better to process shorter jobs at the beginning. We expect that this advantage may be even more significant for more complex cases of simultaneous queries to multiple storages and dynamic priority rebalancing.

4. Results and discussion
This work considers a system of aggregated data collection and access with a metadata catalog and $S$ distributed independent storages. For the case of user requests that address only one of the storages, a system of $S$ parallel queues was proposed. A procedure was proposed for estimating the time it takes to complete individual stages of user request processing, an algorithm for processing user requests by the system was described, a mathematical model of the described process was developed, and the problem was formulated to minimize the user waiting time in the queue $s$. To investigate the possibilities of the optimal solution to this problem, simulation modeling was performed, which has shown the advantage of the chosen Priority method of task ordering compared to FIFO and LIFO approaches. More complex cases of system behavior and other approaches to determine the task priorities will be considered in further studies.

Acknowledgements: The author acknowledges the help of the colleagues from projects Astroparticle Physics Data Storage (APPDS) and German-Russian Astroparticle Physics Data Life Cycle Initiative (GRADLCI), especially to A. Haungs.

References
[1] Mons B, Neylon C, Velterop J, Dumontier M, da Silva Santos L O B and Wilkinson M D 2017 Information services & use 37(1) 49-56
[2] Reiser L, Harper L, Freeling M, Han B and Luan S 2018 Molecular plant 11(9) 1105-08
[3] Lagoze C, Van de Sompel H. 2005 Implementation guidelines for the open archives initiative protocol for metadata harvesting Protocol version 2.0 of 2002-06-14, document version 2005/05/03T22:51:00Z http://www.openarchives.org/OAI/2.0/guidelines.htm
[4] Yiotis K 2005 Information technology and libraries 24(4) 157-62
[5] van Wezel J, Streit A, Jung C, Stotzka R, Halstenberg S, Rigoll F et al. 2012 arXiv preprint arXiv:1212.5596
[6] Bychkov I, Demichev A, Dubenskaya J, Fedorov O, Haungs A, Heiss A et al. 2018 Data 3(4) 56.
[7] Schörner T et al. The PAHN-PaN NFDI Consortium. The binding Letter of Intent of the PAHN-PaN NFDI Consortium https://www.dfg.de/download/pdf/foerderung/programme/nfdi/absichtserklaerungen_2019/2019_pahn_pan.pdf Last visited 05.11.20

[8] Shvachko K, Kuang H, Radia S and Chansler R 2010 The hadoop distributed file system IEEE 26th symposium on mass storage systems and technologies (MSST) 1-10

[9] Zaharia M, Xin R S, Wendell P, Das T, Armbrust M, Dave A et al. 2016 Apache spark: a unified engine for big data processing Communications of the ACM 59(11) 56-65.

[10] Kacsuk P, Kiss T 2007 Towards a scientific workflow-oriented computational World Wide Grid. CoreGRID Technical report https://westminsterresearch.westminster.ac.uk/item/91934/towards-a-scientific-workflow-oriented-computational-world-wide-grid Last visited 05.11.20

[11] Kruse R L 1987 Data Structures and Program Design (2nd ed) (Prentice-Hall, Inc. div. of Simon & Schuster) pp 150