Journal of Physics: Conference Series **664** (2015) 042030 doi:10.1088/1742-6596/664/4/042030

Directory search performance optimization of AMGA for the Belle II experiment

Geunchul Park, Jae-Hyuck Kwak[1], Taesang Huh and Soonwook Hwang
on behalf of the Belle II Computing Group[2]
Korea Institute of Science and Technology Information (KISTI),
245 Daehak-ro, Yuseong-gu, Daejeon, KOREA
[2]https://belle2.cc.kek.jp/~twiki/pub/Public/ComputingPublic/AuthorList4Belle2Computing.tex

E-mail: gcpark@kisti.re.kr

Abstract. AMGA (ARDA Metadata Grid Application) is a grid metadata catalogue system that has been developed as a component of the EU FP7 EMI consortium based on the requirements of the HEP (High-Energy Physics) and the biomedical user communities. Currently, AMGA is exploited to manage the metadata in the gBasf2 framework at the Belle II experiment, one of the largest particle physics experiments in the world. In this paper, we present our efforts to optimize the metadata query performance of AMGA to better support the massive MC Campaign of the Belle II experiment. Although AMGA exhibits very outstanding performance for a relatively small amount of data, as the number of directories and the metadata size increase (e.g. hundreds of thousands of directories) during the MC Campaign, AMGA suffers from severe query processing performance degradation. To address this problem, we modified the query search mechanism and the database scheme of AMGA to provide dramatic improvements of metadata search performance and query response time. Throughout our comparative performance analysis of metadata search operations, we show that AMGA can be an optimal solution for a metadata catalogue in a large-scale scientific experimental framework

1. Introduction

AMGA is a grid metadata catalogue service that was developed through the EGEE project and reflects the requirements of two major communities: the High-Energy Physics community and biomedical research community. Subsequently, AMGA's function was further developed through participation in the EMI (European Middleware Initiative) and EGI (European Grid Infrastructure). Currently it is being utilized at various sites as well as by the Belle II experiment [1].

One of the largest experiments in the world, Belle II utilizes AMGA [2] to process a large amount of metadata. Data generated from the Belle II experiment is processed through the gBasf2 framework [3] using a distributed computing framework called DIRAC (Distributed Infrastructure with Remote Agent Control) and processed files containing the results are stored by distributing them to data centers all over the world. The metadata for the results is stored and managed in AMGA so that researchers can use it to search for desired data files and apply the results to their own research [4][5][6].

[1] Corresponding author.

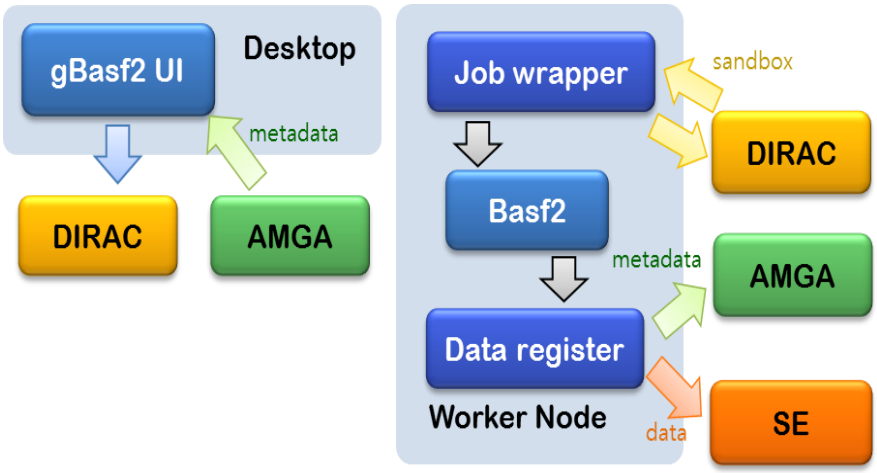

Figure 1. gBasf2 framework and AMGA

2. Belle II MC Production campaign and AMGA Issue

In high-energy physics experiments that process a large amount of data, an MC (Monte-Carlo) production campaign is implemented to check the computing model and distributed computing environment [5][6][7]. The Belle II experiment performed an MC production campaign four times starting from February 2013 to date. Our MC production campaign increased the throughput for each cycle, as shown in Fig 2. However, during the third cycle we encountered slow response times from AMGA and an overload of the AMGA server (Fig 3) [8].

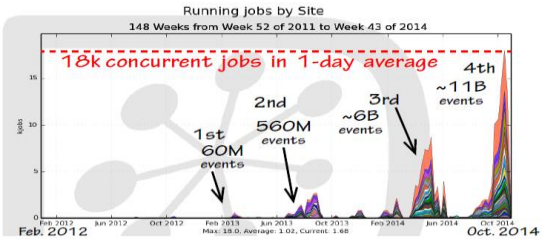

Figure 2. Belle II MC production campaigns

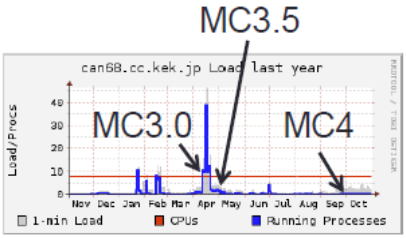

Figure 3. Overload of AMGA server during MC3.0

2.1. Analysis of issue

When the 3rd MC production campaign was performed, the response time of the 'ls [directory]' command took 0.01 seconds at the initial installation, but this slowed the performance down to approximately 1 second. The 'ls [directory]' command is one of the most often used commands, as it contains all entries in the target directory. Due to this issue, a large number of AMGA and PostgreSQL sub-processors were kept running continuously. This resulted in a high system load, and in fact, the memory usage reached its limit. The cause, analyzed with the cooperation of the Belle II framework building team, was found to be the slowness of the AMGA directory search speed.

As the number of directories created in AMGA increased — at that time, more than 300,000 directories were created in the server used in the Belle II experiment — the speed decreased proportionally. The slowdown of response was almost independent of the number of entries; indeed, the analysis results showed that the number of directories governed the response time.

2.2. Bottleneck

If the client runs a command by accessing AMGA, a series of processes is performed, as shown in Figure 4. During those processes, various tests were performed to find the bottleneck point of the

response. As a result, it was found that it took a long time for AMGA to request the query to the database and to retrieve the result set in the postgres sub-processor.

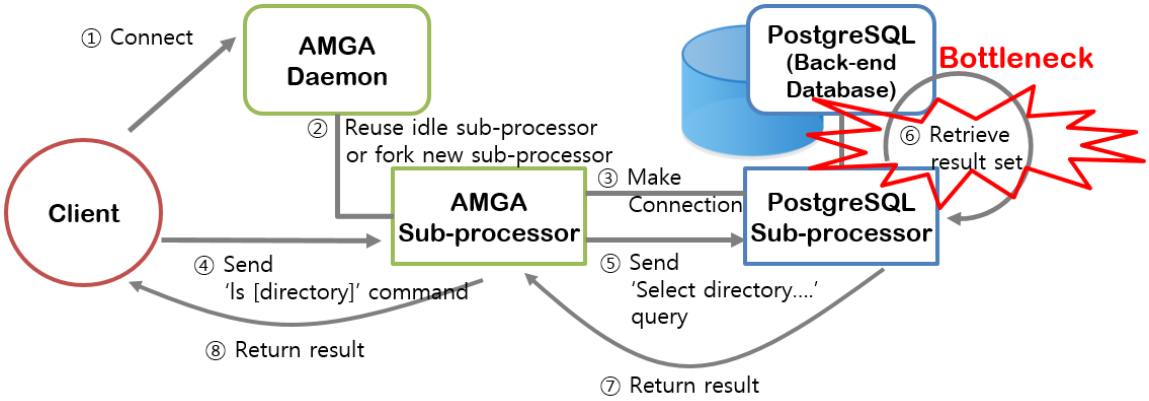

Figure 4. Bottleneck in search workflow

3. Enhancement of directory search performance

This section describes the data storage, search structure and method in AMGA, and presents our modification plan to improve performance.

3.1. Storing and searching directory information in the AMGA

AMGA uses two types of tables to manage directories and entries. There is one masterindex table that stores basic information for all directories, and there are directory tables to store entries belonging to each directory. Whenever a new directory is created in AMGA, it creates a single associated directory table for the new directory and inserts information for this new directory in the masterindex table. Then, entries added to this directory are stored in the previously created directory table.

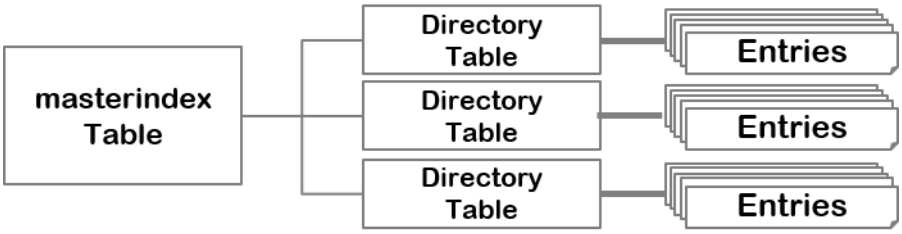

Figure 5. Structure of directory and entry information of AMGA

When reading the content of a directory using the 'ls [directory]' command, AMGA provides the entry and sub-directory information that belong to the given target directory. Entry information is retrieved from the corresponding directory table and sub-directory information is retrieved from the masterindex table, respectively. The bottleneck is not caused by the entry information retrieving part, but rather by the sub-directory information retrieving part. The entry information retrieving part is fast enough despite the huge number of entries, but the sub-directory information retrieving part is slowed down as the number of directories increases.

3.2. Optimization of scheme and directory search query

The reason for the slowdown was the use of an expensive POSIX-style pattern matching expression to retrieve the sub-directory information from the masterindex table instead of a low-cost exact-matching expression. The observed performance issues occurred with this POSIX-style expression because this method is a high-cost operation that compares strings. For string comparison operations, using a

database indexing technique does not improve performance, and so when there are increasing numbers of comparable targets, its performance is seriously degraded.

To solve this problem, we added a parent column, which stored the information of the parent directory to the masterindex table and changed the directory information search method from pattern search to exact-match search. As a result, the search performance was improved significantly.

- AMGA Command
  Query> ls /belle2/user/gcpark
- Old database search query : version 2.4.0
  SELECT “directory", "table_name", "flags", "owner_name", "permissions", "acls”
  FROM public.masterindex
  WHERE **"directory" ~ '^/belle/user/gcpark/[^/]+$'**;
- New database search query : version 2.5.0
  SELECT "directory", "table_name", "flags", "owner_name", "permissions", "acls"
  FROM public.masterindex
  WHERE **"parent" = '/belle/user/gcpark/'**;

4. Test results

As discussed in this section, tests were conducted to determine the extent to which the search performance was improved before and after applying the changes. Additionally, the response time was tested for a large amount of data in a high-end server.

4.1. Test methods

The test program is written in Python, and uses AMGA Python APIs that are included in the Belle II framework. The response time is measured according to the interval between the sending of the 'ls [directory]' command to the AMGA server and the receiving of all the entries’ information. The response time includes the time used to read all of the entries stored in the target directory. This means that the response time can be changed based on the number of entries stored in the target directory. As explained in the following section 4.2, because we used a virtual machine with a small storage space, all of the directories had 100 entries. Subsequently, as noted in section 4.3 and 4.4, all of the directories had 1,000 entries to make the test environment as similar as possible to the real Belle II framework. In all of the tests, the target directory was selected randomly using the ‘ls [directory]’ command.

4.2. Comparison test 2.4.0 VS 2.5.0

To compare the two programs, AMGA versions 2.4.0 and 2.5.0, we ran both of them on identical virtual machines configured with 2 CPUs and 16 GB RAM on an Intel-based PC running Scientific Linux 5 x86 and PostgreSQL 8.1.23. We applied the same schema used in the Belle II experiment for AMGA testing. The tests were carried out by creating 100 entries for each directory and then increasing the number of directories.

The test for measuring the response time was conducted by reading all of the entries stored in the randomly selected directory using the ‘ls [directory]’ command. In this case, each of the directories had 100 entries. Each test was performed 10,000 times, and then the average was calculated, randomly selecting the directory from which to read data.

Table 1. Comparison of response times (seconds) of V2.4.0 and V2.5.0

Directories		1	50	100	500	1K	5K	10K	50K	100K
Entries		100	5K	10K	50K	100K	500K	1M	5M	10M
Response Time(s)	**V2.4.0**	0.0017	0.0032	0.0038	0.0089	0.0139	0.0556	0.1081	0.5222	1.0268
	V2.5.0	0.0017	0.0017	0.0017	0.0018	0.0018	0.0018	0.0018	0.002	0.0026

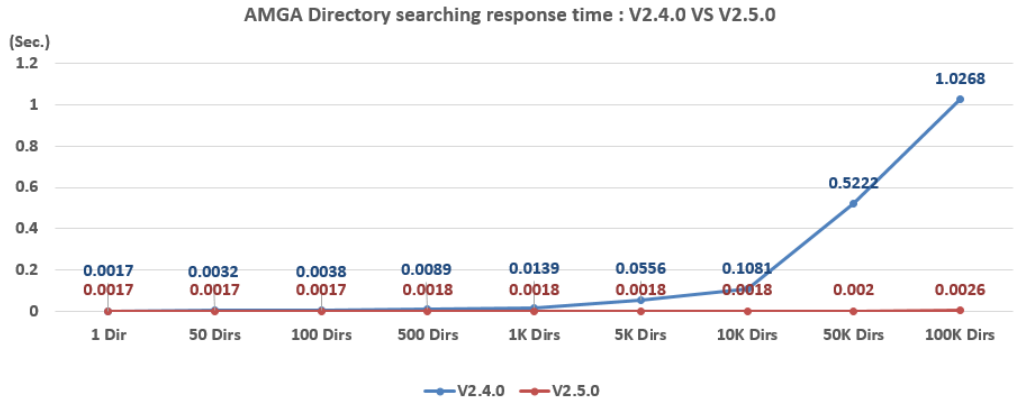

Figure 6. Graph of AMGA directory search response time

The test results confirmed that in the 2.4.0 version with no enhanced search method, as the number of directories increased, the response time slowed considerably at high directory numbers. In contrast, with the new 2.5.0 version, the response time was increased logarithmically as the number of directories increases.

4.3. Mass-data test on high-spec server

To evaluate AMGA performance in an actual environment, we carried out a test using a large amount of data in a high-end server. The specifications of the server used in this test are shown in Table 2.

Table 2. Specifications of server for test

	Specification
CPU	2 x Intel® Xeon® Processor , 3.0GHz 10Core, 25M Cache
RAM	128GB DDR3 ECC RAM 1600MHZ /(24 DIMM slots)
SSD	2 x 120GB SSD
HDD	3TB 7.2K RPM near Line SAS 3.5 * 12EA(RAID 0)
OS	Scientific Linux 5 x86
DB	PostgreSQL 8.1.23

The test was conducted using AMGA 2.5.0 with enhanced performance and Belle II experiment schema. In this test, the same test command 'ls [directory]' was used; however, in contrast to the previous tests, the number of entries read by AMGA for each directory was 1,000, 10 times more than in the previous tests. Once again, the target directory was randomly selected.

Table 3. Disk I/O measurements of SSDs and HDDs

		Buffered disk read (hdparm –t)	Cached disk read (hdparm –T)	Write speed (dd)
Speed (MB/s)	**SSDs**	1,030	12,842	487
	HDDs	1,617	12,463	486

For the purpose of this test, we examined the AMGA performance when using either the SSD or HDD for storage. Table 3 shows the disk I/O measurements. The 'hdparm' and 'dd' commands were used for the measurement. The measurements ran to each SSD and HDD, repeated 30 times. Since the 12EA HDDs are configured as the RAID 0, they are faster than the SSDs in the 'Buffered disk read' column. The SSD test was limited to a maximum of 150M entries in 150K directories due to the limited capacity of the SSD; the HDD was tested with up to 1G entries in 1M directories.

Table 4. Average response time of high-spec server

Directories		10K	50K	100K	150K	200K	500K	700K	1M
Entries		10M	50M	100M	150M	200M	500M	700M	1G
Response Time(s)	**SSD**	0.0149	0.0148	0.0148	0.0148	N/A	N/A	N/A	N/A
	HDD	0.0146	0.0146	0.0143	0.0146	0.0147	0.0146	0.0156	0.0156

In spite of the large number of entries in both the SSD and HDD, the results showed that a rapid response time was maintained. We plan to perform tests of more directories and entries in the future.

4.4. Database shared buffer

Table 5 below shows the test results in detail. The test was performed with data from 30K directories or 150K directories on the SSD and 500K directories on the HDD. The 'ls [directory]' command was run for 1,000 cycles in the randomly selected directory, repeated 30 times. Table 4 shows the test results for the 1st to 7th runs along with the average results for the 8th to 30th runs.

Table 5. Detailed test results for high-spec server

Times		**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**	**Avg. (8th~30th)**
Response Time(s)	**30K**	0.01649	0.01549	0.01509	0.01493	0.01485	0.01483	0.01477	0.01470
	150K	0.01646	0.01542	0.01506	0.01486	0.01484	0.01479	0.01478	0.01477
	500K	0.03363	0.01974	0.01666	0.01459	0.01381	0.01387	0.01395	0.01361

It should be noted that as the number of repetitions increased, the response time was faster, but that after the 7th run, a steady speed was maintained. This result can be attributed to the shared buffer zone in the database. The database has a certain space for the shared buffer in which the result set is stored for later reuse. During the 1st to 6th runs, the result set was stored in the shared buffer, and after the 7th run, the stored result set was transmitted to another shared buffer, so it was possible to have a faster response time. However, since the response time difference between the result set stored in the shared buffer and the one not stored in the shared buffer was not significant, a short basic response time was guaranteed.

5. Conclusions and future work

We analyzed the directory search performance of AMGA, which is used as a metadata catalogue solution in the Belle II experiment. We examined the AMGA process, investigated the cause of an observed performance problem and then developed and applied a performance improvement method and tested the performance of both the original and enhanced AMGA.

In the future, it will be necessary to test with more directories and entries in preparation for the actual Belle II experiment. Additional testing under load generated by a large number of clients is required as well as optimization of the AMGA parameter settings and database for enhanced performance.

6. Acknowledgements

We are grateful for the support and the provision of computing resources by CoEPP in Australia, HEPHY in Austria, McGill HPC in Canada, CESNET in the Czech Republic, DESY, GridKa, LRZ/RZG in Germany, INFN-CNAF, INFN-LFN, INFN-LNL, INFN Pisa, INFN Torino, ReCaS (Univ. & INFN) Napoli in Italy, KEK-CRC, KMI in Japan, KISTI GSDC in Korea, Cyfronet, CC1 in Poland, NUSC, SSCC in Russia, SiGNET in Slovenia, ULAKBIM in Turkey, UA-ISMA in Ukraine, and OSG, PNNL in USA. We acknowledge the service provided by CANARIE, Dante, ESnet, GARR, GEANT, and NII. We thank the DIRAC and AMGA teams for their assistance and CERN for the operation of a CVMFS server for Belle II.

References

[1] Koblitz B, Santos N, and Pose V 2008 *Journal of Grid Computing* **6** 61–76
[2] Ahn S et. al. 2010 *Journal of the Korean Physical Society* **57** issue 4 715
[3] Hara T 2014 *Proceedings of Asia-Pacific Advanced Network* **38** 115-122
[4] Hara T 2014 *Computing for Belle-2* (Tsukuba University, Japan, 3-8 March)
[5] Hara T “Computing at the Belle-II experiment” CHEP2015 proceedings
[6] Miyake H “Belle II production system” CHEP2015 proceedings
[7] Kuhr T 2014 *J. Phys. Conf. Ser.* **513** 032050
[8] Miyake H 2014 *Belle II distributed computing with DIRAC* (CERN, Zurich, 26-28 May)