ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 4, pp: (157-161), Month: October - December 2014, Available at: www.researchpublish.com

Distributed Data Caching For Big Data

N. Sundaram[1], R. Saranya[2]

[1] Assistant Professor, [2]M.E., (CSE),Third Semester, Maha Barathi Engineering College, Vasudevanur, India.

Abstract: ***In*** **this big data world we find daily based data which are generated day by day e.g. google, amazon, Facebook etc. This large amount of data is un-reliable to store, manage and analyze. This large volume of data runs on Commodity hardware, which is parallel arranged.**

The data that are large in volume takes more time to execute for particular process and causes fail to distributed system because these huge volume data runs on distributed system.

To overcome this issue a framework was proposed called HADOOP for big data processing and is being used now days in organizations. It process large amount of data in small amount time rather than large distributed systems. It has automatic fault tolerance capacity to tackle with failed of systems in execution or processing time using Map Reduce programming technique.

Here execution time is still an issue for delivering large amount of data and processing it repeatedly for particular oracle. Existing HADOOP system does not have any feature to reduce time for recompilation.

We propose a HADOOP distributed system for large data processing, which has two type of cache memory one is local cache, and other is distributed centralized cache memory. We use these cache memories for reducing the recompilation time.

I. INTRODUCTION

Big data is a buzzword, or one can say it's a catch-phrase, which can be used to describe a huge volume of structured, unstructured, text, images, audio, video, log files, emails, simulations, 3D models, military surveillance, e-commerce and so on that is so massive that it's difficult to process using traditional database and software techniques.

In most enterprise scenarios the data is too big or it moves too fast or it exceeds current processing capacity. Big data is nothing but a synonym of a huge and complex data that it becomes very tiresome, difficult or slow to collect, store, sort, process, retrieve and analyze it with the help of any existing relational database management tools or traditional data processing techniques.

Big Data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process the data within a tolerable elapsed time.

II. CHARACTERISTICS OF BIG DATA

Data scientists break big data into four dimensions: volume, variety, velocity and veracity.

Volume: BIG DATA depends upon how large it is. It could amount to hundreds of terabytes or even petabytes of information.

Velocity: The increasing rate at which data flows into an organization.

Variety: A common theme in big data systems is that the source data is diverse and doesn't fall into neat relational structures.

Big Data Problems

In existing they use local cache for execution that is not efficient for faster execution. It can only use cache in the local systems.

Volume: As an example, Terabytes of posts generated on Facebook or 400 billion annual twitter tweets could mean Big Data! This enormous amount of data will be stored somewhere to analyze and come up with data science reports for different solutions and problem solving approaches.

Velocity: Big data requires fast processing. Time factor plays a very crucial role in several organizations. For instance, millions of records are generated in the stock market which needs to be stored and processed with the same speed as its coming into the system.

Variety: There is no specific format of Big Data. It could be in any form such as structured, unstructured, text, images, audio, video, log files, emails, simulations, 3D models, etc. Until now we have been working with only structured data. It might be difficult to handle the quality and quantity of unstructured or semi-structured data that we are generating on a daily basis.

III. HOW BIG DATA HANDLES THE ABOVE PROBLEMS

The motivation over this system is to use intermediate results for further execution of the applications, where we do not want to execute results which are previously processed. Thus in this system we proposed a distributed cache strategy which keeps previously processed data in cache memory at local and centralized. So there are two cache memories will be used for faster execution of applications, one will be local cache and another will be centralized distributed cache.

Distributed File System (DFS): In DFS, we can divide a large set of data files into smaller blocks and load these blocks into multiple number of machines which will then be ready for parallel processing. For example, if we have 1 Terabyte of data to read with 1 machine and 4 Input / Output channels with each channel's reading speed id 100MB/sec, the whole 1 TB data will be read in 45 minutes. On the other hand, if we have 10 different machines, we can divide 1 TB of data into 10 machines and then the data can be read in parallel which reduces the total time to only 4.5 minutes.

Parallel Processing: When data resides on N number of servers and holds the power of N servers, then the data can be processed in parallel for analysis, which helps the user to reduce the wait time to generate the final report or analyzed data.

Fault Tolerance: The Fault tolerance feature of Big Data frameworks (like Hadoop) is the one of the main reason for using this framework to run jobs. Even when running jobs on a large cluster where individual nodes or network components may experience high rates of failure, BigData frameworks can guide jobs toward a successful completion as the data is replicated into multiple nodes/slaves.

Use of Commodity Hardware: Most of the Big Data tools and frameworks need commodity hardware for its working which reduces the cost of the total infrastructure and very easy to add more clusters as data size increase

Big Data Key Characteristics

1. Cache memory is easy to configure
2. It is reliable and scalable
3. Recompilation is more faster than fist compilation
4. Maintenance cost is less because of Map Reduce programming technique.
5. Low memory is required to allocate cache memory.

Scope of Big Data

The scope of the Distributed HADOOP cache memory is to accelerate HADOOP cluster and faster execution. It can be used large scale development of large amount of data. The centralized distributed cache memory is used for the scope.

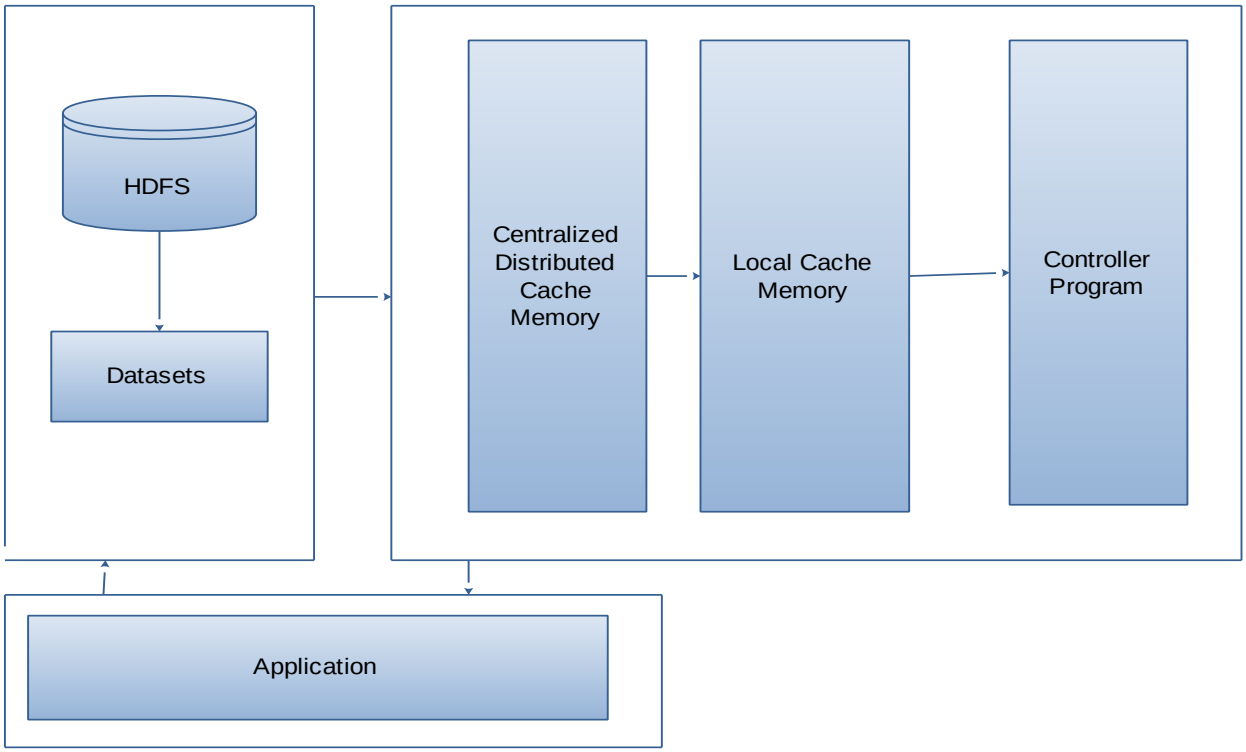
In this world, data has to grow based on user's activity and growing organizations, all organization keeps there data on commodity hardware or rack, which are easy to add or remove for future installments. Thus, this dynamic or scalable feature of HADOOP is useful in organization.

The data in commodity hardware are dynamic and needs to analyze for future consideration of user data. Thus here developer needs to write there code and have to run application, but application has to run whole data and process then need to give output or result. It consumes more time. If developer needs to do any changes or correction in code and need to recompile, it takes much time and more time spends in recompilation.

Thus, we need a system, which can minimize the recompilation time as it in JSP-servlet features ,Where the servlet recompilation is much lesser than first compilation.

Our objective here is to reduce recompilation time using local cache and distributed centralized cache memory which is present on master and slave machine.

Proposed Architecture



About Centralized Cache Manager Scheme

Centralized cache management in HDFS is an explicit caching mechanism that allows users to specify paths to be cached by HDFS. The Name Node will communicate with Data Nodes that have the desired blocks on disk, and instruct them to cache the blocks in off-heap caches.

Centralized cache management in HDFS has many significant advantages.

1. Explicit pinning prevents frequently used data from being evicted from memory. This is particularly important when the size of the working set exceeds the size of main memory, which is common for many HDFS workloads.

2. Because Data Node caches are managed by the Name Node, applications can query the set of cached block locations when making task placement decisions. Co-locating a task with a cached block replica improves read performance.

3. When block has been cached by a Data Node, clients can use a new, more-efficient, zero-copy read API. Since checksum verification of cached data is done once by the Data Node, clients can incur essentially zero overhead when using this new API.

4. Centralized caching can improve overall cluster memory utilization. When relying on the OS buffer cache at each Data Node, repeated reads of a block will result in all 'n' replicas of the block being pulled into buffer cache. With centralized cache management, a user can explicitly pin only m of the n replicas, saving n-m memory.

Modules

1. **Configuring Hadoop Cluster**

- We need to configure JDK in LINUX system.
- Install Open-SSH Server for Hadoop network.
- Disable IPV6.
- Extract and copy HADOOP tar into LINUX system.
- Configure Hadoop xml files.

2. **Upload datasets to HADOOP distributed file system**

- Developer formats namenode using "HADOOP namenode -format" command.
- Developer creates a directory under user folder based on username.
- Create a directory to upload input data..
- Upload data to input directory.

3. **Configuring Local Cache Memory**

- Allocate cache memory for every slave node.
- Activate Local cache memory.
- Recompiling Code

4. **Configuring distributed cache memory**

- Allocate cache memory at master node.
- Activate cache memory for distributed system.

5. **Set Timestamp**

- Set start time for controller program.
- Set end time for Controller Program.
- Calculate the difference between timestamp.

Technical requirements

Software Requirements for End Users

- **Operating System:** Linux OS with distributed Environment.

Software Used in Development

- **Platform:** Java
- **Technologies used:** Java, Hadoop
- **Database:** HDFS
- **Development environment (IDE):** Eclipse

IV. CONCLUSION

We present the design and procedures of Distributed cache for data searching framework in Map Reduce.

We implemented distributed cache in Hadoop for data searching to save compilation time. Map Reduce technique coding will execute the searching of data's in efficient way. so, system processing speed will be improved and compilation time will be saved.

Future enhancement

Speculative execution of searching data to reduce compilation time

REFERENCES

[1] Amazon web services. http://aws.amazon.com/.

[2] Cache algorithmes. http://en.wikipedia.org/wiki/Special:Search/Cache_algorithms.

[3] Google compute engine. http://cloud.google.com/products/computeengin

[4] Hadoop. http://hadoop.apache.org/.

[5] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. of ACM, 51(1):107–113, January 2008.

[6] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. ACM Comput. Surv.,35(2):114–131, June 2003.

[7] B. Brown, M. Chuiu and J. Manyika, "Are you ready for the era of Big Data?" McKinsey Quarterly, Oct 2011, McKinsey Global Institute

[8] C. Bizer, P. Bonez, M. L. Bordie and O. Erling, "The Meaningful Use of Big Data: Four Perspective – Four Challenges" SIGMOD Vol. 40, No. 4, December 2011

[9] D. Boyd and K. Crawford, "Six Provation for Big Data" A Decadein Internet Time: Symposium on the Dynamics of the Internet and Society, September 2011, Oxford Internet Institute

[10] D. Agrawal, S. Das and A. E. Abbadi, "Big Data and Cloud Computing: Current State and Future Opportunities" ETDB 2011, Uppsala, Sweden