Dynamic Block Replication

Ankita Girish Wagh
Sambit Mishra

CSCE 662 Class Project

Outline

- Problem Formulation
- Basics
- State of Art
- Main Ideas
- Proposed Solution
- Evaluation
- Conclusions

Problem Formulation

- HDFS 782- In a large and busy cluster, a block can be requested a number of times by many clients leading to hot blocks.
- The number of concurrent connections on a datanode is fixed(default 256) so that datanode does not run out of memory.
- If more number of clients access concurrently, they have to retry connection leading to delay [time lag].
- If their retry count increases more than the threshold, they have to wait for some time before trying again leading to more delay.

Basics

- In Current HDFS the number of replicas for all blocks of a file is same.
- This replication is configurable from hdfs-site.xml. This factor is generally 3.
- Block placement policy - HDFS keeps 3 replicas on different data nodes out of which 2 data nodes are on the same rack and 1 on different rack.
- Whenever DFSClient has to read or write a block, it asks the address to the namenode.
- Namenode gives the address of all the 3 replicas.
- DFSClient tries to access the data from the nearest datanode.

Basics Cont...

- Once the DFSClient gets the metadata, ie. the address of data nodes, it caches it.
- If the nearest datanode cannot be accessed, it will try the next replica to read or write data.
- At any point of time, its difficult for the namenode to guess whether a block has become a hot block i.e. large number of clients are trying to access it concurrently.
- Data nodes send block report periodically which is a list of blocks kept on them.
- Name node processes this report and updates its table accordingly.

State of Art

- Currently data nodes send in their heartbeat messages the concurrent connections it is serving.
- Dynamic replication is still an open issue in apache community .
- The number of retries by the client, the number of concurrent connections and number of replicas are configurable but they don't presently address the issue of hot blocks.

Main Ideas

- There are a number of ways to address this issue-

- File Replication- Everytime a client asks for a block we can increase the counter by 1. We can keep a threshold of concurrent clients above which we can increase the file replication.

Advantage- Simple implementation as namenode already has the information of the file.

Disadvantage-We do not precisely know which block is the hot block because once the address is sent the client will cache it. There is no way to know number of clients accessing the block.

How do we decrease the replications??

Cont..

- Datanode Replication-

When the number of concurrent connections on a datanode reach the threshold, namenode can ask datanode to replicate all the blocks on it.

Advantage-Namenode knows the value of concurrent connections from the heartbeat messages. Secondly when the concurrent connections are more the probability of the block being hot block increases.

Disadvantage- It is possible that the block responsible for so many concurrent connections is just one but we will end up replicating all the blocks present on that node from different files.

Proposed Solution:Block Level Replication [Algorithm]

- If in the block report apart from the list of blocks we add the active connections, Namenode will know exactly how many clients are accessing which blocks at that time.
- So we added a parameter of active connections in the block class.
- We process the block report at namenode to extract the connections.
- For everyblock we calculate the sum of connections from all replicas .
- If the sum is greater than the threshold we replicate the block.

Experimental Evaluation and Test Setup

- Our aim is to test whether dynamic replication can reduce the read time of blocks.
- Our test bed consisted of 2 datanodes having 1 rack , replication factor 1 and concurrent connections threshold as 3.
- We spawned concurrent threads of various number and issued "cat command" on hadoop file system to read the file .
- We compared the timings of read with and without algorithm .

Solution...

10 Threads

Without Algorithm	With Algorithm
25413	24571
25733	25861
26052	26049
26792	26127
27067	26149
27302	26338
27950	26347
28018	26361
28391	26690
24913	23416

15 Threads

Without Algorithm	With Algorithm
40041	36922
41112	37426
41137	38022
41497	39050
42782	39223
43123	39490
44750	39509
45006	39508
45232	39535

45261	39725
45422	39785
45610	40002
45924	40014
46136	40044
41137	39020

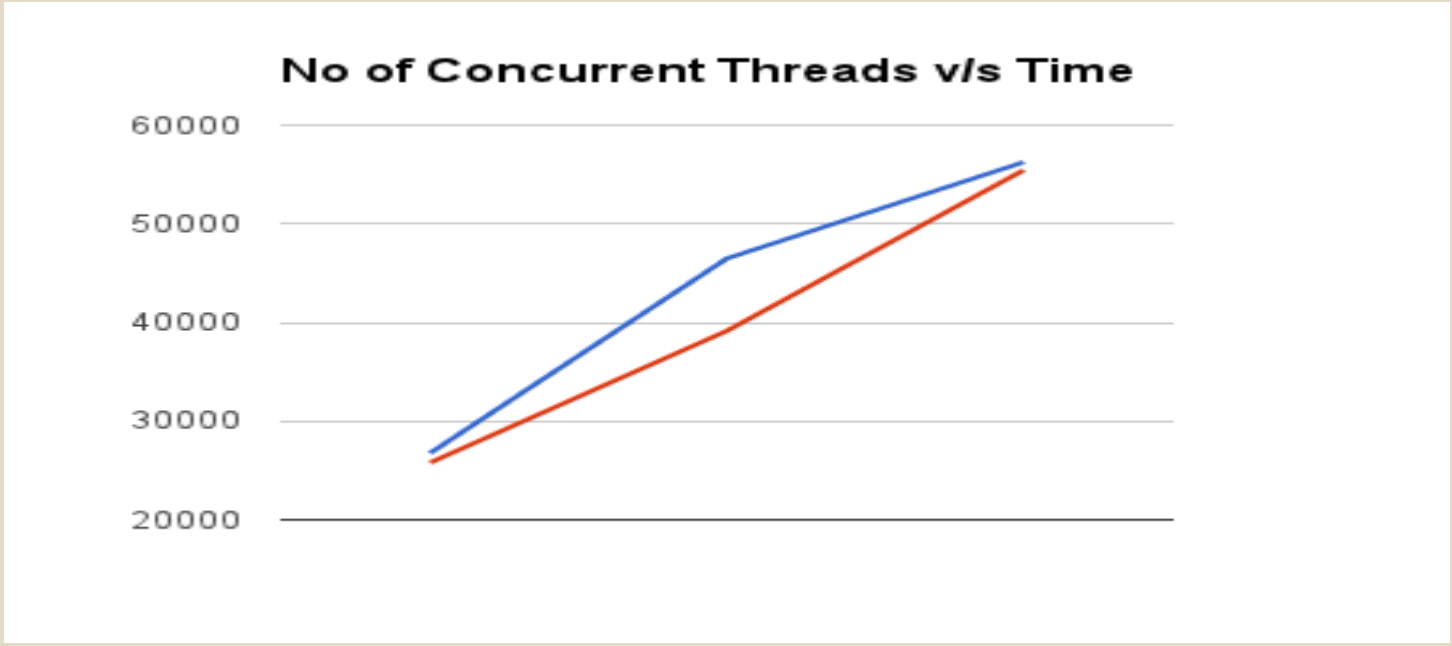
Solution Cont...

20 Threads:

Without Algorithm	With Algorithm
54227	53977
54892	54363
54940	54350
56121	55318
57029	55726
57833	56035
58016	56388
58071	56383
58159	56434
58615	56473

59238	56553
59414	56619
59945	56810
59991	56987
60243	57007
60282	57069
60504	57106
60766	57132
60845	57109
62505	57185

Average Time to Read :



Blue Bar : Without Algorithm

Orange Bar : With Algorithm

Results

The sudden dip in improvement from 15 to 20 threads can be because of using a single machine.

No. of threads	Without Algo	With Algo	Improvement
10	26763.1	25791	3%
15	46487.8	39151.6	15%
20	56251	55437	1.4%

Constraints & Challenges

- Due to unavailability of infrastructure, we couldn't test this algorithm on real world cluster with more than 256 concurrent connections.
- The CPU computation speed is shared between all the datanode replications as our test set was on single machine. We think testing on multi machine will lead to better results.
- Instead of read we can test our algorithm with more complex computations.

Conclusions

- We tried to solve the problem of Dynamic Replication at "block level". The results show that it enhances efficiency by reducing the time of reads.
- Since the number of replications have increased,the clients will not have to retry on a single block to read data.
- We don't need to replicate the entire file for one hot block.

Future Improvements:

- Replication based on threshold of concurrent connections per replica on a data node.
- Able to choose the datanode where the replica should be placed.
- Experimentally determining the threshold of time limit and then decreasing the extra copies instead of right away deleting them when connections decrease.

References:

- http://hadoop.apache.org/docs/stable/single_node_setup.html
- http://wiki.apache.org/hadoop/GettingStartedWithHadoop
- http://docs.oracle.com/javase/6/docs/api/overview-summary.html
- https://issues.apache.org/jira/browse/HDFS-782

Questions ??