Distributed System Report

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—The abstract goes here.**

I. Failure Handling

A. Failure Detection

In our distributed file system, the master maintains a timestamp table to record the last heartbeating time for every registered slave. Slaves are designed to send heartbeat messages to the master every t_b seconds. After receiving the heartbeat message, the master updates the timestamp of that slave to the current system time. On the master side, a thread is built to check the timestamp table every t_c seconds. Once the absolute difference between the timestamp of a slave and current system time of master exceeds a prescribed threshold, the master adds the slave to a *dead_slave* list. After checking the timestamp of all slaves, the master starts the recovery process if the *dead_slave* list is not empty.

B. Data recovery: Master Side

In the data recovery function, the master first establishes a list containing all damaged files and their damaged part indexes, according to the *dead_slave* list and the metadata of all files. For each damaged file, the recovery process is given as follows.

- Input: Damaged indexes set S, e.g., $\{O2, O1O2, O3O4\}$;
- Initialization: Mark damaged indexes as LOST, and set their recovery set R as empty set (following the example, $O2$ is marked as LOST and $R_{O2} = \emptyset$); Mark non-damaged indexes as OK and set their recovery set R to a set containing themselves ($O1$ is OK, and $R_{O1} = \{O1\}$ meaning O1 can be recovered by itself, or no recovery needed);
- Bottom-up Recovery Phase: Check indexes in a bottom-up order: $O1, O2, O3, O4, O1O2, O3O4, O1O2O3O4$. If an index i is LOST and both of its children, say $c1$ and $c2$, are OK, mark i as OK and update R_i to $R_{c1} \cup R_{c2}$; In our example, $O3O4$ is recovered and $R_{O3O4} = R_{O3} \cup R_{O4} = \{O3, O4\}$ while $O2$ (no child) and $O1O2$ (child $O2$ is LOST) cannot be restored.
- Top-down Recovery Phase: Check indexes in a top-down order: $O1O2O3O4, O1O2, O3O4, O1, O2, O3, O4$. If an index i is LOST and both of its father f and sibling s are OK, mark i as OK and update R_i to $R_f \cup R_s$; In our example, $O1O2$ is recovered first by $O1O2O3O4$ and $O3O4$ with $R_{O1O2} = R_{O1O2O3O4} \cup R_{O3O4} = \{O3, O4, O1O2O3O4\}$. Then $O2$ gets recovered by $O1O2$ and $O1$ with $R_{O2} = R_{O1O2} \cup R_{O1} = \{O1, O3, O4, O1O2O3O4\}$.
- Return: If there exists empty R_i for any index i, return failed, else return all R_i with $|R_i| \geq 2$.

If the recovery process returns failed, which means we cannot recover this file by remaining parts, our system delete this file from file list. Otherwise, we assign a new slave (with best-effort distributed law) for each missing index, and call the recovery process at the slave side with input argument: index to recover and a list of $(index, slave)$ pair for every needed index.

C. Data recovery: Slave Side

Upon receiving the recovery request from the master, the slave obtain the missing index for the damaged file and know where to obtain the needed indexes to recover the missing index with the help of input arguments. Afterwards, the slave do the recovery with the following steps:

- Obtain indexes for recovery: The recovery slave read needed index from the slave which owns the index for the preparation of recovery. Here, the recovery slave acts like a client, and do a read operation from the slave who owns the index. For example, if the input argument is $O3O4$, and $< (O3, slave2), (O4, slave3) >$, the recovery slave will do a read operation to $O3$ from slave2, and a read opeartion to $O4$ from slave3 for the preparation of recovery index $O3O4$.
- Recovery data: After obtaining the indexes needed to recover the missing index, the recovery slave do the encode process for the indexes read in the previous step. For the example in previous step, the recovery slave is able to encode for $O3O4$ with the content of index $O3$ and $O4$. After encoding, we obtain the content of index $O3O4$.
- Write to stable storage: The recovery slave write the content of missing index into its local disk and finish the recovery process for this missing index.

If there are many recovery requests from the master, the slave will do recovery for missing indexes one by one until it finishes all of the recovery request. When the slave doing recovery, it will not block the read and write requests from clients.

D. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.