**Distributed Operating System (COP-5615)**

**Project – 3a Report**

***Prepared by:***

**Ankit Jain (UFID - 66934055)**

**Saumya Agarwal (UFID: 73539184)**

***Introduction:***

This project concentrates on saving and accessing tuples on servers by using client-server distributed model. The server is capable of handling requests from multiple clients using multiple threads and each server can connect and communicate with other servers in a distributed environment. All servers concurrently maintain a single, atomic tuple space for the clients. The clients can connect to multiple servers from its CLI interface. Tuple space provides a repository of tuples that can be accessed concurrently by the producer and consumer clients. The servers use the synchronization system to process the insertion and deletion operations as atomic operations. For example, if two clients add tuples at the same time, both must be found in the tuple space when both have finished. Likewise, if two clients send templates that match the same tuple, and they both request deletion, then only one of the two clients shall receive the matching tuple. Also the project handles the race conditions and failure conditions in which if one server stops/fails, then the remaining servers remains unhindered and are still able to function with the tuples they have. The project also aims to make the system more secure, robust and scalable.

***Security***

The system is made secure by implementing an authentication system which saves the authentic user’s information on the server side. Whenever a new server is started, a new administrative password is given to it. Users having this administrative password can create and delete users. Only the authorized user having the correct server password can access the server and will be able to query the tuple space. The unauthorized user will be denied access to the tuple space.

***Robustness:***

The tuple space is not just stored in existing data structures on each server but also additionally stored in the local file system. The files are accessed only when there is an insert or delete operation to the server. Match query will only be done on the data existing in the local data structure. This highly reduces the number of read and writes operations on the disk and makes the system more efficient resulting in faster query response. Thus the tuples space persists even if the server stops or fails during operation. The system can also be made fault-tolerant by a mechanism in which if one node fails/becomes inactive, it is replaced by another node to forward the query.

***Scalability:***

We use the hierarchical routing protocol for sending the tuple information from one server to another. There will be parent-child relationship between the active servers. The search query is propagated from the queried server to its parent and child servers which forward the query to their connected servers. So the search query is forwarded only to a limited number of servers at a time.

**References:**

1. Corradi, A, Zambonelli, F, Leonardi, L:“A Scalable Tuple Space Model for Structured Parallel Programming”, IEEE Explore, Programming Models for Massively Parallel Computers, 1995
2. Monnet, S., Morin, C., Badrinath M : “A Hierarchical Checkpointing Protocol for Parallel Applications in Cluster Federations”, Proceedings of the 18th International Parallel and Distributed Processing Symposium
3. Ganesh, V :"Design principles of Scalable, Distributed System"

(<https://www.ibm.com/developerworks/mydeveloperworks/blogs/theTechTrek/entry/design_principles_of_scalable_distributed_system5>)

1. Subramaniyan, R., Raman, P., George, A.D., Radlinski, M.: "GEMS: Gossip-Enabled Monitoring Service for Scalable Heterogeneous Distributed Systems", Cluster Computing, Volume 9, Number 1, January 2006