**Streaming of applications for Rapid FIFO on the multiple cores to implement efficient memory mapping and data processing**

**By:-**

Shivakrishna Bujunoori

**Faculty Advisor** – Mike Litman @awesomefat

**Co-Authors:** Curcio Joshua M, Emo Brian S, Haxton David C, Huesmann Timothy R, Nguyen, Alexander N, Wildauer Timothy

**Faculty Advisor** – Mike Litman @awesomefat

**Co-Authors:** Curcio Joshua M, Emo Brian S, Haxton David C, Huesmann Timothy R, Nguyen, Alexander N, Wildauer Timothy

**Research Statement**

Developing a strategic methodology to solve task scheduling and inter connectivity problems in the multiple core environments to overcome synchronization and scheduling costs in the existing distributed and parallel computing mechanisms.

**Abstract**

As traditional programming languages assumes single instruction code of stream and monolithic memory for processing of structured instructions even though scientists implemented wide range of cores and processor technologies. For example, I7 IBM has 9 cores; Microsoft XBOX 360 CPU has 3 cores and etc.., the problems with the parallelism can be overhead on the communication and synchronization between the outputs. In the areas of distributed environments, data processing and ability to synchronize the tasks between the machines make a lot of sense than single streamed instructions into a single machine which takes lot of execution time. The tedious task in the application programming and data processing in the distributed environments is that synchronization of the sub tasks and also the amount of scheduling costs.

In this research topic I would like to explore the task parallelism in which different actors or certain programs can be implemented in an object oriented language to establish the task of streaming chunked applications into different cores with respect to the guide of a superior program for establishing parallel assignment and collaboration across the machines different levels of cores. The Implementation of filter programs for scheduling the tasks in distributed environments in which each core to attain effective parallel communication and synchronization. By adapting to the individual program counters of the assigned sub program with respect to the processors available in the multi-processing environments effects the execution of the entire program drastically as we are following the approach of divide and conquer in application level. These task parallel filters enable the sub program to use full structure of the core independently. The granularity of the task scheduling depends completely on the implemented filter programs than usage of parallelism.

**Overview:**

Distributed Applications are sophisticated software applications that run over a network of computers within the network at the same time. These commonly shared resources in the assigned jobs can lead to have effective job schedulers and also task schedulers among network nodes in this case they widely distributed across the globe and considerable into a single cluster. For maintaining these cluster environments to perform a specific task using features of cluster computing within the paradigm of divide and conquer computing resources in the cluster. The drawback of using distributed applications is that if there is a node failure in the processing of a job can lead to failover of other nodes in the distributed environment. Using Streaming of Applications from a different environment probably running in cloud to be able to access the current states of each node in the cluster by tracking certain process or job status which will be useful in the point of failure of node’s assigned process by introducing the Software as a Service application monitoring schedulers in the cluster node typically running in a web browser.

**Existing System:**

In a data processing job which consists of huge data manipulations over a cluster may be considered as tedious job with respect to task schedulers in the clustered computing application. The Distributed application insisted to run on a point of fail over state which means it can either be completed at one shot or none. If there is a synchronization inconsistency across the nodes over conquering results of the subtasks may lead to in efficient and inconsistent states of data changes over shared resources. Consider the case of Data Migration between Relational database management servers which is a costly operation involving client real time feeds and also valuable reporting feeds of any business.

**Problems in Existing System:**

* Tedious task of synchronization among parallel distributed environments
* Single stream of instruction execution even though multiple core environment exists.
* Problems of synchronization of sub tasks and amount of scheduling costs in the distributed environment
* Point in time failure causes the total loss on investment of a Job in terms of resources and time.
* Data inconsistency over period of time.
* Additional cost for either of the cases such as Redoing of Distributed application over cluster or Data recovery strategy implementation.

**Suggested Methodology:**

Task scheduler application development in which scheduler tracks all of the assigned tasks and their running status details in separate secured server than as part of cluster in large dataset’s data processing. Implementation of filter programs for scheduling these tasks in distributed environment to attain effective parallel communication among participants and synchronization. Using streaming applications where necessary applications were streamed independently to the metadata server instead of request-response strategy in centralized server concept. Instead of centralized server strategy, distributing the whole centralized copies into replicas in different metadata servers will solve issues of centralized server down and etc. To implement these applications to act upon the cluster environment is a cost effective solution which will be a great solution compared to Data recovery strategies or running distributed queries again on the cluster. These additional tracking applications holds each cluster participants metadata in terms of execution participant details, configuration details, level of involvement tracking to overcome connectivity problems.

**Uses over Existing System:**

* Avoidance of centralized server down problems.
* Inter connectivity problems among participants in distributed computers.
* Delay reductions in case of server lost in distributed environment with ability to accurate track of task status.
* Continues connection maintenance using streaming applications than request-response strategy. For example AWS – applications were streamed to support as an on-demand strategy than desktop applications.
* Reducing in costs as using service oriented strategy than product licensing.

**Practical Scenario Example:**

For example, migration of an RDBMS data to the NOSql using distributed computing. The problem scenario – intermediate node’s state lost in computing lead to restart the node and redo the entire sub process or the whole process of the migration which may cause due to network issues, hardware mal function or failure of synchronization by resulting nodes to inability to participate into the computing from the point of contact lost.

By following the Idea of this research paper, the following steps will contribute to overcome this particular problem of synchronization and intermediate node loss problem. Keeping streamed application task scheduler having access to the log files of each read operation from the source end and also destination. Keeping estimated input lengths which means for each node in the network accessible to task scheduler. For example, in migration use case 10 tables for each node to migrate out of 100 tables along with relationships in the network. The natural applied software component in the task scheduler applications does has the feature of Pre query optimization in the case of failure event to be use full for recovery. This Pre Query optimization enables the task scheduler to scan the sub process query accordingly to provide save points for keeping the job inline.

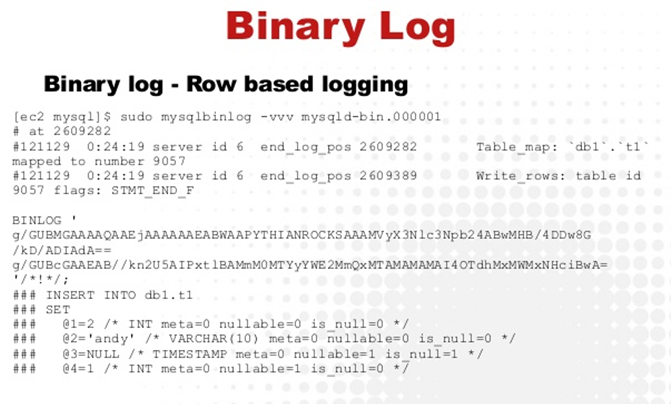
**Use case Implementation:**

* Reading log records of destination database for maintaining consistency with the source database.
* Pre-defined sets of save points with respect to write operations in the destination database like save points in tables wise and for every 1000 records of a table.

**Pre-Conditions**:

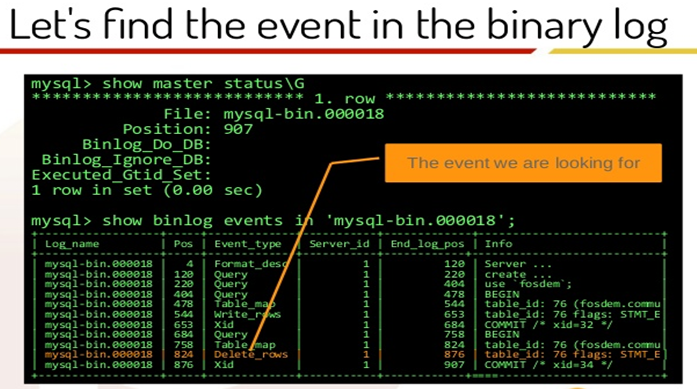
* All tables should have a unique key to distinguish records and records should be migrated either in ascending or descending order of that unique key.
* In the destination database, the last write operation can be filtered and decide the record belong to which set of a save point junction. To redo the case of failure- It would be easy to establish the optimized query with limit and sorting options in sql. The node starts exactly at the point of failure step.

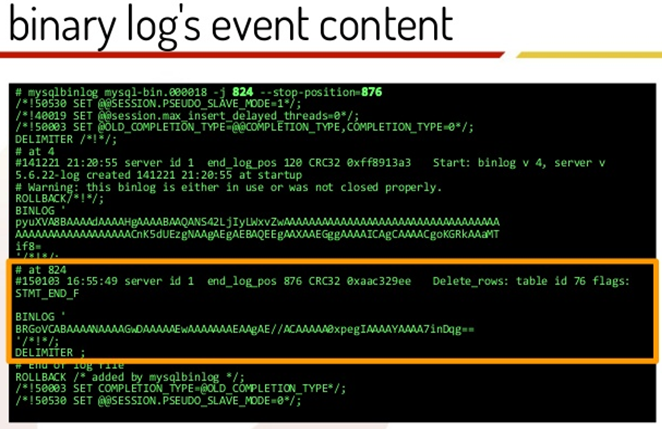
**Sample screen of bin-log file in Mysql Server:-**



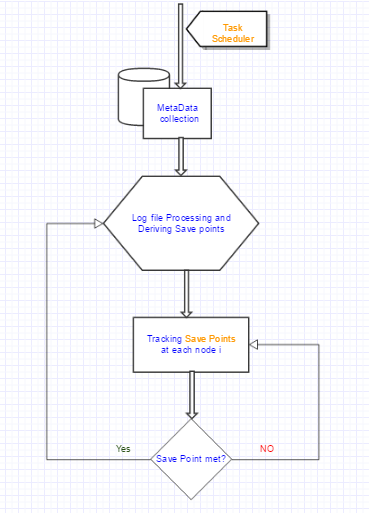
Screen flow to identify delete operation from Events captured binlog file Mysql:-







**Suggested Save points capturing flow in the problem of Migration:**

****

In each node, mostly in a web browser, the task scheduler runs and tracks the log file based on the flow suggested above to keep track of save points. Sample save points may be, save point record for every table migration to destination or every 1000 records from huge table in source to destination. In case of intermediate node loss, once the node restarts and based on the information available in the task scheduler, migration query will be processed accordingly to serve the rest of the process of migration.

**Pros:**

* Disaster recovery using effective task schedulers.
* Time saving strategy in the middle of huge computations in the case of using group of computers.
* Able to implement effective parallel programs to implement in multi core and cluster environments.
* Effective communication between tasks in the multi core environment
* Flexible communication and data exchange over network of different configurations with multiple cores.
* Synchronization of parallel tasks within programs on clusters like neural networks and in heavy industries which needs huge amount of data processed in parallel environments.

**Cons:**

* Third party task scheduler operates in the remote area than in the current environment.
* Expensive to maintain.

**Future Enhancements:-**

* Extensive growth for task schedulers in case of modular components gets lost in the distributed environments.
* Building sophisticated scheduling applications pertaining to solve neural network problems where there exists the distributed computing of huge data.
* Medical, statistical analysis tracking application development.

**References:**

* 1. Michael I. Gordon, W. T. (2008). Exploiting Coarse-Grained Task, Data,and Pipeline Parallelism in Stream Programs. <https://www.lri.fr/perso/~waller/cours/fr/articles/2008/cohen2.pdf> .
  2. Steve J. Chapin, Syracuse University. DISTRIBUTED AND MULTIPROCESSOR SCHEDULING <https://www-users.cs.umn.edu/~jon/papers/handbook.pdf> .
  3. Helen D. Karatza, Parallel Job Scheduling in Homogeneous Distributed Systems <http://www.cs.cmu.edu/~harchol/WORMS04/people/karatza/Karatza2.pdf>.
  4. Mohsen Bayati (2007). Iterative Scheduling Algorithms. <http://web.stanford.edu/~balaji/papers/07iterativescheduling.pdf>
  5. M. (n.d.). Guide to Scaling Web Databases with MySQL Cluster. Retrieved December 18, 2016, from <https://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/>
  6. M. (n.d.). A Guide to MySQL Enterprise Monitor. Retrieved December 18, 2016, from <https://www.mysql.com/why-mysql/white-papers/a-guide-to-mysql-enterprise-monitor/>
  7. How can I parse mysql bin log file for statements (Row/Mixed Format). (n.d.). Retrieved December 18, 2016, from <http://dba.stackexchange.com/questions/105269/how-can-i-parse-mysql-bin-log-file-for-statements-row-mixed-format>
  8. S. (2016). Shyiko/mysql-binlog-connector-java. Retrieved December 18, 2016, from <https://github.com/shyiko/mysql-binlog-connector-java> .
  9. How to Read MySQL Binary Logs. (2008). Retrieved December 18, 2016, from <https://egloo.wordpress.com/2008/11/19/how-to-read-mysql-binary-logs/> .