IJDCST @October Issue- V-1, I-6, SW-39
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Fine Tune Updates on Data Streaming Using False Tolerant EDF

K.Chaitanya Kumar [1], Dr J.Srinivas Rao[2]

[1]Student, Nova College of Engineering and Technology, Ibrahimpatnam, Krishna Dist, Andhra Pradesh, India

[2] Professor, Nova College of Engineering and Technology, Ibrahimpatnam, Krishna Dist, Andhra Pradesh, India

Abstract: Data streaming manage systems is a computer program to manage continuous data streams. For this process present in the streaming of object relation data management system applications. Traditionally developed applications are new arrived application status on data ware house streaming applications. In these applications we are using Earliest Deadline First scheduling algorithm for developing dynamic update on data streaming in ware houses. Due to the burden of the overhead presented in EDF on data warehouse server, it will take more complexity for solving streaming in data warehouse server. So in this paper we propose to extend our existing work for supporting dynamic updates in earliest data steaming applications. We propose FTEDFS; it is the acronym for Fault Tolerant Earliest Deadline First Scheduling technique. In this technique we ignore unspecified results of the every user's content and theirs usable message forward to other user present in the network. Our experimental results show efficient data steaming process and give efficient results generation of data construction in data ware housing.

Index Terms: Time-redundancy, real-time scheduling, fault-tolerance, Data warehouse maintenance, online scheduling uniprocessor embedded systems, earliest deadline- first.

I. INTRODUCTION

A DSMS also offers a flexible query processing so that the information need can be expressed using queries. However, in contrast to a DBMS, a DSMS executes a *continuous query* that is not only performed once, but is permanently installed. Therefore, the query is continuously executed until it is explicitly uninstalled. Since most DSMS are data-driven, a continuous query produces new results as long as new data arrive at the system. A data warehouse is database used for reporting and data analysis. It is central repository of data which is created by integrating data from one or more disparate sources. Data warehouses store current as well as historical data and are used for creating trending reports for senior management reporting such as annual and quarterly comparisons.

IJDCST @October Issue- V-1, I-6, SW-39
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

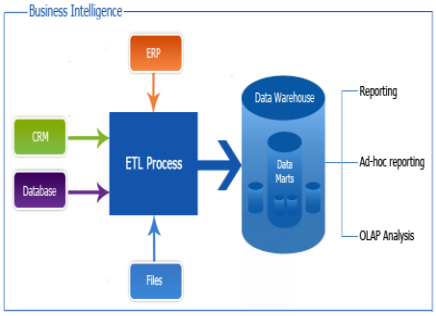

Figure 1: Data ware housing process with efficient result generation.

The data stored in the warehouse are uploaded from the operational systems (such as marketing, sales etc., shown in the figure to the right). The data may pass through an operational data store for additional operations before they are used in the DW for reporting. The typical ETL-based data warehouse uses staging, data integration, and access layers to house its key functions. The staging layer or staging database stores raw data extracted from each of the disparate source data systems.

We then propose a scheduling framework that handles the complications encountered by a stream warehouse: view hierarchies and priorities, data consistency, inability to preempt updates, heterogeneity of update jobs caused by different inter arrival times and data volumes among different sources, and transient overload. A data warehouse constructed from an integrated data source system does not require ETL, staging databases, or operational data store databases. The integrated data source systems may be considered to be a part of a distributed operational data store layer. Data federation methods or data virtualization methods may be used to access the distributed integrated source data systems to consolidate and aggregate data directly into the data warehouse database tables. Transient faults in real time systems are generally tolerated using Time Redundancy applications present in our dynamic data streaming.

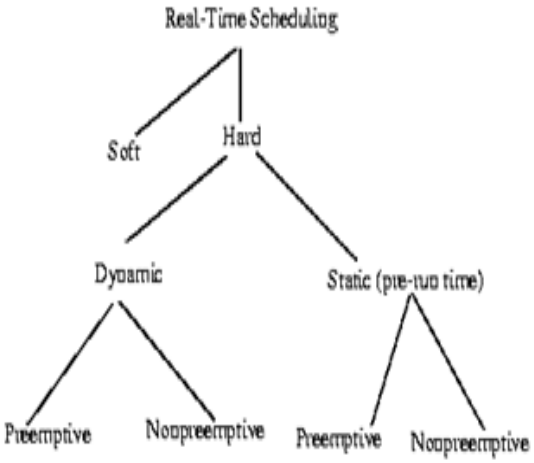

Figure 2: Data streaming textual taxonomy independent assurance.

One important issue in real-time embedded systems is the scheduling of tasks in these systems. Modern real time scheduling research has mostly concentrated on generating efficient algorithms for guaranteeing that tasks meet their deadlines without considering faults.

By using fault tolerant issues present in our proposed work criteria. Our proposed technique should be used to develop efficient data streaming on data ware housing applications. Real time system process the efficient data accessing for the behavior of the data streaming applications present in the data warehousing process.

II. BACKGOUND WORK

This enables a real-time decision support for Business-critical applications that receive streams of append-only data from external sources.

• Online stock trading, where recent transactions generated by multiple stock exchanges are compared against historical trends in nearly real time to identify profit opportunities.

• Credit card or telephone fraud detection, where streams of point-of-sale transactions or call details are collected in nearly real time and compared with past customer behavior.

• Network data warehouses maintained by Internet Service Providers (ISPs), which collect various system logs and traffic summaries to monitor network performance and detect network attacks. For this application development traditional used data processing techniques are as follows: Earliest Deadline First algorithm for processing real time data efficiency between every user present in our network. In this paper we propose to develop the Fault Tolerant Earliest Deadline First algorithms/technique is clear to describe the scheduling process present in the data ware housing specifications. Using our proposed work we will develop the performance of the steaming present in our ware house construction process.

In fact for any pair of average task utilization and mean time to failure (α, MTTF) the efficient time redundancy is added to the schedule. An event driven simulator is designed and implemented for performance evaluation. Moreover, this scheme can be applied to any non-fault-tolerant scheduling policy for preemptive and periodic tasks (e.g., Rate-Monotonic (RM) scheduling policy).

Our proposed work gives efficient data transmission between every user aspect present in the data ware housing applications.

III. EXISTING APPROACH

The traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. Where traditional data warehouse store layers of complex materialized views over terabytes of historical data. Traditionally developed techniques are accessed in the present situations of the data streaming process of the every user data. In this we are planned to data processing using scheduling data of particular user aspects in data ware housing applications.

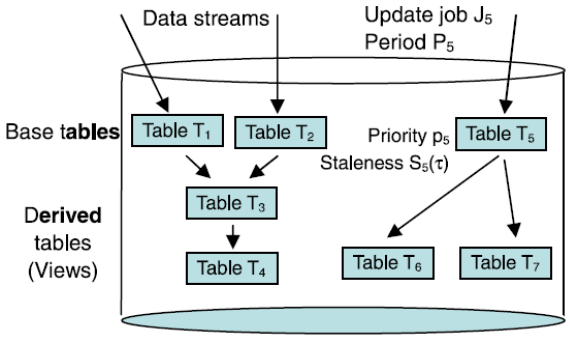

Figure 3: Data streaming process using EDF algorithms.

Each data stream i is generated by an external source with a batch of new data present in the data ware housing applications. An important data process of traditional data streaming is, it access information due to the insufficient data streaming process present in our proposed work.

IV. PROPOSED APPROACH

The general approach to fault-tolerance in uniprocessor systems is to make sure there is enough slack in the schedule to allow for re-executing of any task instance, if a fault occurs during its execution.

Tasks are executed following the usual EDF scheme if no faults occur (the slack is not used). However, when a fault occurs in a task, a recovery scheme is used to re-execute that task. In the EDF policy with utilization bound less than 100%, there is a natural amount of slack in uniprocessor.

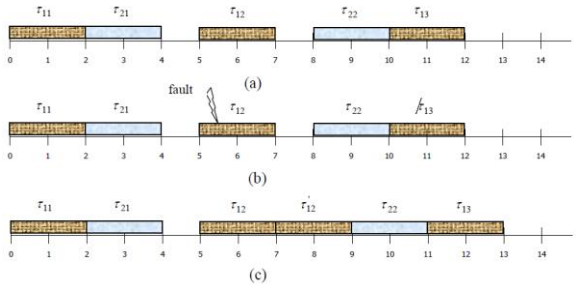

Figure 4: (a) tasks are scheduled in not present faults, (b) τ 12 has encountered with a fault, (c) the recover scheme has been called and τ 12 has been re-executed using time-redundancy.

Our proposed work can be efficiently data streaming of every user present in data ware housing applications.

V. PERFORMANCE RESULTS

In this section we describe the efficient data streaming ware housing applications. The proposed data warehouse stores integrated information from multiple distributed data sources. In effect, the warehouse stores materialized views over the source data. The problem of ensuring data consistency at the warehouse can be divided into two components: ensuring that each view reflects a consistent stare of the base data, and ensuring that multiple views are mutually consistent. Guarantying multiple view consistency (MVC) and identify and define formally three layers of consistency for materialized views in a distributed environment.

For example data streaming as follows:

Example: consider two tasks with $c1 = 2, T1 = d1 = 5$, $c2 = 2$, $T2 = d2 = 8$. The two utilizations are $2 / 5 = 0.4$ and $2 / 8 = 0.25$ respectively. Assuming that backup utilization is 0.2, i.e.,$UB = 0.2$. (3.2), gives us a bound of 0.8 while the sum of utilizations of the task is 0.65. Since $0.65\ 0.8\ 1 = = \leq - = = \Sigma$ *FT EDF N i i U U u* , the tasks are schedulable. In figure 1.a no fault has occurred while the tasks are executed. In figure 4.b fault has occurred when τ 12 (τ 12 denotes the second period of τ 1) was being executed and has been detected at time 7, then recovery scheme has been called, i.e., new copy of τ 1 has been added to the schedule (figure 4.c), and has been completed using the slack time (time redundancy). By increasing the value of time redundancy has a considerable scheduling process.

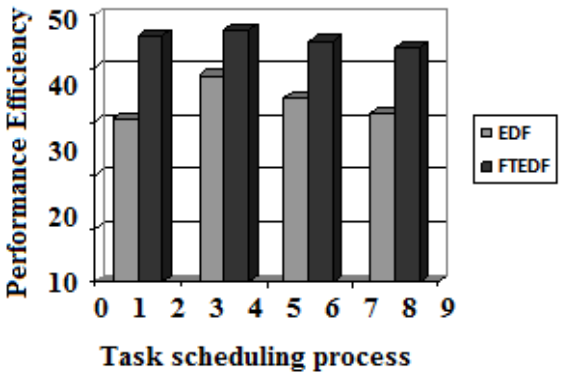

Figure 5: Comparison results of the existing EDF and our proposed FTEDF.

As shown in the above diagram we are showing data streaming with different data aware applications.

VI. CONCLUSION

In this paper we propose to access the services of the data streaming applications using efficient data

transfer. For this application development traditional development applications are Earliest Deadline First streaming applications. But these abstraction processes are the taking efficient data streaming applications based on time invariant with feasibilities. Dynamic scheduling is online and uses scheduled by test to determine whether a set of tasks can meet their deadlines. The present paper talks about static and dynamic scheduling algorithms and operating systems support for these mechanisms. Efficient data transmission between every user present in our proposed work data accessing can be done with different data properties. Our experimental result shows the data efficiency with increasing of the data processing between real time systems specifications. As further improvement of our proposed work, in our proposed we mainly concentrate on the data transferring with low time redundancy process. We plan to extend to our proposed in developing of "Granularity" with increasing efficient data streaming process in ware house applications.

VII.REFERENCES

[1] Lukasz Golab, Theodore Johnson, and Vladislav Shkapenyuk," Scalable Scheduling of Updates in Streaming Data Ware Houses", IEEE Transactions On Knowledge And Data Engineering, Vol. 24, No. 6, June 2012.

[2] Hakem Beitollahi, Seyed Ghassem Miremadi2, " Fault-Tolerant Earliest-Deadline-First Scheduling Algorithm", 1-4244-0910-1/07/$20.00 ©2007 IEEE.

[3] R. Al-Omari, A. K. Somani, G. Manimaran, "Efficient overloading techniques for primary-backup scheduling in real-time systems", Journal of Parallel and Distributing Computing, 64 (2004) 629–648, March. 2004.

[4] D. Mossé, R. G. Melhem, S. Ghosh, "A No preemptive Real-Time Scheduler with Recovery from Transient Faults and Its Implementation", IEEE Trans. Software Eng., vol.29, no.8, pp. 752-767 , 2003.

[5] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N.-E. Frantzell, "Supporting Streaming Updates in an Active Data Warehouse," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 476- 485, 2007.

[6] M. Sharaf, P. Chrysanthis, A. Labrinidis, and K. Pruhs, "Algorithms and Metrics for Processing Multiple Heterogeneous Continuous Queries," ACM Trans. Database Systems, vol. 33, no. 1, pp. 1-44, 2008.

[7] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling Updates in a Real-Time Stream Warehouse," Proc. IEEE 25th Int'l Conf. Data Eng. (ICDE), pp. 1207-1210, 2009.