# AN ANALYSIS OF DISTRIBUTED REAL TIME SYSTEMS: AN OVERVIEW

**M.S.Khatib[1], Professor Dr. Mohammad Atique[2]**

*[1]Department Of Computer Science & Engineering,*

*Anjuman College Of Engineering & Technology, Sadar, Nagpur, M.S.(India)*

*[2]P G Department Of Computer Science, S.G.B. Amravati University, Amravati, M.S.(India)*

## ABSTRACT

*Different distributed real-time systems (DRS) must handle aperiodic and periodic events under diverse sets of requirement. A real time distributed computing has heterogeneously networked computers to solve a single problem. So co-ordination of activities among computers is a complex task and deadlines make it more complex. The performance of the system depends on many factors such as traffic workloads data base system architecture, first configurable component middleware services for admission control and load balancing of aperiodic and periodic event handling, underlying processor, disk speeds, concurrency control, transaction management. Simulation study has to be performed to analyze the performance under different transaction scheduling, different workloads arrival rate priority policies, altering slack factors and preemptive policies.. The throughput of the system depends on the arrival rate of transaction. The performance can be enhanced by adaptively minimizing the slack factor.*

***Keywords*: *Real Time Systems, Distributed Threats, Earliest Deadline First, Rate Monotonic Analysis, Utility Accrual (UA) Scheduler, First Come First Served Scheduling Scheme, Weighted Round Robin Scheduling Scheme, EDF Scheduling Scheme, Rate Based Scheduling Scheme.***

## I. INTRODUCTION

Consider a real-life scenario, wherein the buffer space (memory) available at the sinks (required for holding and processing the loads) varies over time, and the loads have deadlines and propose efficient "pull-based" scheduling strategies with an admission control policy that ensures that the admitted loads are processed, satisfying their deadline requirements. The design adopts the divisible load paradigm, referred to as the divisible load theory, which is efficient in handling large volume loads in situations where in processing for several divisible (partition able) loads need to be completed within their respective deadline requirements, while the processing nodes have finite capacity constraints. Divisible loads are a class of loads that require homogeneous processing and can be partitioned into arbitrary smaller fractions. These load portions, which bear no dependence relationships among themselves, can then be assigned to individual nodes for processing.

The analytical flexibility offered by the divisible load theory is thoroughly exploited to design resource-conscious algorithms that make the best use of the available resources in a cluster. Both interleaving and non interleaving techniques to process tasks (jobs) that are admitted into the system are employed.

The problem of scheduling large-volume loads (divisible loads) within a cluster system, which is part of a grid infrastructure is envisage, this cluster system as a cluster node comprising as set of computing nodes [1][2][3]. Communication is assumed to be predominant between such cluster nodes and is assumed to be negligible

within a cluster node. The underlying computing system within a cluster can be modeled as a fully connected bipartite graph comprising sources, which have computationally intensive loads to be processed (very many operations are performed on them) and computing elements, called sinks, for processing loads (data). This represents the fact that each source can schedule its load on all the sinks.

In real-life situations, one of the practical constraints is satisfying the deadline requirements of the loads (arriving in real time form multiple source nodes) to be processed while taking into account the availability of the buffer (memory) resources at the sinks nodes, since the memory available at the processing nodes to store the received load and process them is limited and is overlooked with Buffer Estimation Strategy .

## II. LITERATURE REVIEW

In this section various approaches for Distributed real-time systems are reviewed. They are as specified below.

### 2.1 Middleware Services With Configurable Strategies

Several important challenges are services must be able to provide configurable strategies, and configuration tools must be added or extended to allow configuration of those strategies, the specific criteria that distinguish which service strategies are preferable must be identified, and applications must be categorized according to those criterion. Appropriate combinations of services strategies must be identified for each such application category, according to its characteristic criteria.

#### 2.1.1 Distributed Real Time System Characteristics

job Skipping means that some jobs of a task are executed while other jobs of the same task may not be admitted. Overhead Tolerance depends on an application's specific overhead constraints. State Persistency means that states are required to be preserved between jobs of a same task. For proportional control systems task are stateless and only require current information, so jobs can be reallocated dynamically. Components Replication depends on an application's throughput requirements[4][5][6]. Replication is used here to reduce latency through load distribution, not for fault tolerance purposes. Only those applications with replicated components can support task reallocation, whereas those that cannot be replicated, due to constraints on the locality of sensors or actuators cannot support task reallocation.

#### 2.1.2 Admission Control Strategies

Admission control offers significant advantages for systems with aperiodic and periodic tasks, by providing online schedulability guarantees to tasks arriving dynamically. It supports two different strategies: AC-per Task and AC-per Job.

#### 2.1.3 Idle Resetting Strategies

A job remains in the current set even if it has been completed, as long as its deadline has not expired. Therefore, the use of the resetting rule can remove the contribution of completed sub jobs earlier than the deadline. No-IR, IR-per-Task, IR-per-job

#### 2.1.4 Load Balancing Strategies

Use a heuristic algorithm to assign subtasks to processors at runtime, which always assigns a subtask to the processor with the lowest synthetic utilization among all processors on which the application component corresponding to the task has been replicated. No-LB, LB-per-Task, LB-per-Job.

## 2.2 A Database backlog Estimation Technique

A fine-grained closed loop admission control based on the backlog model and incoming load smoothing. Backlog estimation and control-theoretic approaches aim to support the desired service delay bound without degrading the data freshness, critical for real-time data services. Specifically, the design, implementation, and evaluation is based on two feedback controllers based on linear control theory and fuzzy logic control theory, to meet the desired service delay. Workload smoothing, under overloads, helps the database admit and process more transaction in a timely fashion by probabilistically reducing the burstiness of incoming data service delay and throughput. A closed-loop admission control and probabilistic load smoothing schemes considerably outperform several baselines in the experiments undertaken in a stock trading database test bed . Fine grained admission control techniques based on the relation between the estimated backlog and service delay. In which linear control theory, fuzzy logic control theory and a hint-based scheme for smoothing incoming workloads is applied.

2.2.1 Fuzzy Control of Service Delay

A fuzzy logic admission controller is designed to directly model and controlled a nonlinear relation between the backlog and data service delay.

2.2.2 Key Ideas for Rule based Design

Different linguistic values to characterize both the fuzzy input and output.

## 2.3 A Slack Management Technique

The Service-Rate-Proportionate (SRP) Slack Distribution, for real-time distributed embedded systems to reduce energy consumption. It is considered with EDF and Rate-Based scheduling schemes that are most commonly used with embedded systems [7][8]. A fault-tolerant mechanism has also been incorporated into the proposed technique in order to utilize the available dynamic slack to maintain checkpoints and provide for rollbacks on faults.

Furthermore, as reliability and dependability become important in such distributed embedded systems, there is also an important consideration to provide fault tolerance in the system. The incorporation of fault tolerance leads to an overhead on the use of slack for maintaining checkpoints and rollback in such distributed embedded systems. In real-time system designs, Slack Management is increasingly applied to reduce power consumption and optimize the system with respect to its performance and time overheads. This Slack Management Techniques exploits the idle time and slack time of the system schedule by frequency/voltage scaling of the processing elements in order to reduce energy consumption. It introduces a dynamic slack management technique for heterogeneous distributed embedded systems to reduce power consumption. It presents a static slack distribution heuristic to be used for task admittance and demonstrates how task criticality can be handled for hard real-time systems. It demonstrates the effectiveness of the SRP slack distribution technique with the dynamic and rate based scheduling schemes. The different components are

2.3.1 Busy Interval

These intervals help determine the time for the application of Dynamic Voltage Scaling (DVS) or Dynamic Power Management (DPM) at a given node. DVS is applied in the busy intervals and DPM is applied during the idle intervals. The busy interval is determined from the specifications of applications inputs.

2.3.2 Worst-case Delay and Traffic Descriptor

The "worst-case delay' defines the upper bound on the delays experienced at nodes in the path of a task set . It is represented as a vector for all of the nodes in the path of the given task set. Since we consider heterogeneous processing elements, the experienced worst-case delays are different for the same task set at different nodes in the distributed systems.

2.3.3 Periodic Service Rate Determination

The online slack management technique that takes advantage of the runtime variations of the executing task sets heavily depends on the service rate at a given node. The service rate is evaluated at the start of every interval. Such a "periodic service rate" (PSR) determination is a dynamic mechanism which operates on feedback information from the traffic descriptor of a given node for a given interval. First, the new service rate should guarantee the processing of the tasks in the upcoming interval by their delay bounds. Second, this service rate must guarantee the processing of the unprocessed tasks that were left in the queue (and that arrived during the previous intervals) by their worst-case delay bounds. Last, the new service rate must lie within the peak service rate bound of that processing element.

2.3.4 Dynamic Scheduling Scheme

As the task set inputs become during system operation, employing static scheduling schemes cannot help in obtaining the maximum runtime benefits in meeting task criticality requirements and providing for optimal energy savings. Hence, for a general-purpose model design, the inclusion of dynamic scheduling schemes (for examples, EDF and RBS) is essential for design completeness. Between the two, EDF is the more efficient and widely used dynamic scheduling scheme.

### 2.4 More-less Approach

A design approach which maintains the freshness of temporal data while reducing the CPU workload incurred by periodic sensor transactions. It is applied to multiprocessor systems. Transactions are considered in a given priority order and their period and deadlines are assigned. An important issue is to determine the priority order so that the CPU workload imposed by transaction can be minimized. Shortest validity first is an efficient assignment order to minimize the CPU workload for update transactions [9].

In this approach no concurrency control is considered for sensor transactions. It is assumed that sensors always sample the value of a temporal data at the beginning of its period and all the first instances of sensor transactions are initiated at the same time. However a sensor transaction generated by that sensor may arrive at a real-time database with an arrival latency which is also referred as jitter. The period of a sensor transaction is assigned to be more than half of the validity interval of the temporal data updated by the transaction.

### 2.5 One-One Approach

In this approach the period and relative deadline of a sensor transaction have to be equal to the data validity interval. Because the separation of the execution of two consecutive instances of a transaction can exceed the validity interval data can become invalid under the on-one approach. So this approach cannot guarantee the freshness of temporal data in real-time database systems.

### 2.6 Half-Half Approach

In order to guarantee the freshness of temporal data in real-time database systems the period and relative deadline of a sensor transaction are each typically set to be less than or equal to one half of the data validity

interval the farthest distance based on the arrival time of periodic transaction instance and the finishing time of its next instance of two consecutive sensor transactions. Unfortunately even though data freshness is guaranteed this design approach at least double CPU workload of the sensor transaction in the real-time database system compared to the on-one approach [10][11].

**Table 1. Comparison of three approaches**

| Approach | CPU workload | Constraints | Priority of transactions | Validity of Real Time objects |
|---|---|---|---|---|
| One - One | High | No Constraints | FCFS | Guaranteed |
| Half –half | High | No Constraints | Not fixed | Guaranteed |
| More-less | Low | Validity, Deadline, Schedulability. | Shortest validity first | Not Guaranteed |

## 2.7 Privacy-preserving data publishing

Focuses on relational data: in this context, the objective is to enforce privacy-preserving paradigms, such as k-anonymity and l- diversity, while minimizing the information loss incurred in the anonymizing process (i.e. maximize data utility). Existing techniques work well for fixed-schema data, with low dimensionality. Nevertheless, certain application requires privacy-preserving publishing of transactional data (or basket data), which involve hundreds or even thousands of dimensions, rendering existing methods unusable [12]. Two categories of novel anonymization methods for sparse high-dimensional data. The first category is based on approximate nearest-neighbor (NN) search in high-dimensional spaces, which is efficiently performed through locality-sensitive hashing (LSH). In the second category, two data transformation that captures the correlation in the underlying data: reduction to a band matrix and Gray encoding-based sorting. These representations facilitate the formation of anonymize groups with low information loss, through an efficient linear-time heuristic. NN-search yields superior data utility compared to the band matrix transformation, but incurs higher computational overheads. The data transformation based on Gray code sorting performs best in terms of both data utility and execution time.

However, sensitive personal information may be disclosed in this process, due to the existence in the data of quasi-identifying attributes (QID), such as age, zip code, etc. An attacker can join the QID with external information, such as voting registration lists, to reidentify records.

Existing privacy-preserving techniques focus on anonym zing personal data, which have a fixed schema with a small number of dimensions. Through generalization or suppression, existing methods prevents attackers from reidentifying individual records.

However, anonymization of personal data is not sufficient in some applications.The two representations for transactional data which take advantage of data sparseness, preserve correlations among items and arrange transactions with similar QID in close proximity to each other. The first method relies on transformation to a band matrix format, whereas the second employs sorting with respect to binary reflected Gray codes.

### 2.7.1 Multistep processing

It is commonly used for nearest neighbor (NN) and similarity search in applications involving high-dimensional data and /or costly distance computations. Today, many such applications require a proof of result correctness.

In this setting, clients issue NN queries to a server that maintains a database signed by a trusted authority. The server return the NN set along with supplementary information that permits result.

### 2.8 Dynamic Compositions Of Interleaved Tasks

Ensuring reliable transactional processing of Web services is crucial for the success of web service-based B2B and B2C applications. But the inherent autonomy and heterogeneity of Web services render the applicability of conventional ACID transaction models for Web services far from being straightforward. Current Web service transaction models relax the isolation property and rely on compensation mechanisms to ensure atomicity of business transactions in the presence of service failures. However, ensuring consistency in the open and dynamic environment of Web services, where interleaving business transactions enter and exit the system independently, remains an open issue, an architecture that supports concurrency control on the Web services level. An extension to the standard framework for Web service transactions is proposed to enable detecting and handling transactional dependencies between concurrent business transactions. An optimistic protocol for concurrency control that can be deployed in a fully distributed fashion within the proposed architecture. [13][14].

### 2.9 Service –Oriented Computing (SOC)

SOC is becoming the mainstream development paradigm of applications over the Internet, taking advantage of remote independent functionalities. The cornerstone of SOC's success lies in the potential advantage of composing services on the fly. When the control over the communication and the elements of the information system is low, developing solid systems is challenging. In particular, developing reliable web service compositions usually requires the integration of both composition languages, such as the Business Process Execution Language (BPEL), and of coordination protocols, such as WS Atomic Transaction and WS-Business Activity. Unfortunately, the composition and coordination of web services currently have separate languages and specifications. First Identify the major requirements of transaction management in Service-oriented systems and survey the relevant standards. Second, propose a semiautomatic approach to integrate BPEL specifications and web service coordination protocols, that is implementing transaction management within service composition processes, and thus overcoming the limitations of current technologies [15][16]. The systematization of requirements is the starting point for an analysis of current standards and technologies in the field of web services. To propose a framework for the integration of BPEL with transaction protocols such as WS-Atomic Transaction and WS-Business Activity. use the drop-dead order (DDO) one .

2.9.1 The Drop Dead Order

The drop dead order examples describes a scenario where a customer wants to order products from a distributor with the requirements that these must be delivered before the drop-dead date. To satisfy such a request, the distributor will try to find a supplier that has the products available. If found, he will search for a carrier that is able to deliver the products before the drop-dead date. If both the supplier and the carrier are able to fulfill the demands of the customer, the distributor will report to the customer that he can fulfill the order. After the customer has sent a confirmation of the order to the distributor, the latter sends a confirmation to order example in the context of the automotive industry Long lived and concurrent transactions not only traditional transactions which are usually short and sequential. Distributed over heterogeneous environments greater range of transaction types due to different types of business processes, service types, information types, or product flows.

## III. COMPARISON OF DIFFERENT APPROACHES

| Approach | Time Constraint | Flexible Configuration | Periodic/ Aperiodic Task | Open/ Close System | Feedback Control Scheduling | Uncertainty Resource Constraint | Over utilizing System Resources |
|---|---|---|---|---|---|---|---|
| Middleware services | End to end | lacks | both | open | no | no | no |
| Database Backlog | End to end | no | both | closed | yes | no | no |
| Distributed Threads | runtime | yes | both | open | yes | no | yes |
| IPAC Framework | runtime | yes | Either or | closed | yes | yes | yes |
| Slack Management | End to end | yes | aperiodic | open | yes | yes | yes |
| RADIS | yes | no | both | open | yes | yes | yes |
| PPDP | End to end | yes | periodic | open | yes | yes | yes |
| Multistep Processing | End to end | yes | periodic | open | yes | no | no |
| Dynamic Composition of Interleaved Tasks | runtime | yes | both | open | yes | yes | yes |
| Service Oriented Computer | End to end | yes | yes | open | yes | yes | yes |

## IV. ANTICIPATION OF RESULTS.

This section summarizes the culmination of the literature survey carried out in order to put forth the necessity for carrying out the proposed research work. It is proposed :To analyze existing policies and approaches for transaction management in real-time systems. It is also proposed to analyze the performance under different transaction scheduling condition such as different workloads, arrival rates, CPU priority policies altering slack factors and preemptive policies. Design and implementation of new policies for transaction management in real-time systems.

## V. CONCLUSION

A series of simulation study have to be carried out to study the performance of the system under different transaction management situations such as different work load, distribution policies, execution mode- Parallel and distribution, impact of dynamic slack factors to throughput. Proper scheduling of transactions under different priority policies are to be put forward. The different methods of scheduling of task are discussed with regard to time constraint, configuration, periodic/aperiodic task, open/close system, feedback control scheduling, uncertainty resource constraint and over utilization of system resources. The novel approach with regard to adaptive management of transactions with reference to above parameters is to be analyzed.

## REFERENCES

[1] Ming Xiong,Song Han, Student, Kam-Yiu Lam, and Deji Chen, "*Deferrable Scheduling for Maintaining Real-Time Data Freshness: Algorithms, Analysis, and Results",* IEEE Transaction In Computers, vol – 57, No.7, July 2008.

[2] Yuanfang Zhang, Cristopher D. Gill, and Chenyang Lu, *"Configurable Middleware for Distributed Real Time Systems with Aperiodic annd Periodic Tasks*", IEEE Transaction on Parallel and Distributed Systems, Vol. 21, No. 3, March 2010.

[3] Kyoung-Don Kang, Yan Zhou, and Jisu Oh,*"Estimating and Enhancing Real-Time Data Service Delays: Control-Theoretic Approaches",* IEEE Transaction on Knowledge and Data Engineering, Vol 23, No.4 April 2011.

[4] Subrata Acharya, and Rabi N. Mahapatra, *"A Dynamic Slack Management Technique for Real-Time Distributed Embedded Systems*", IEEE Transaction on Computers, Vol. 57 No.2 , February 2008.

[5] Rabi N. Mahapatra and Wei Zhao, *"An Energy-Efficient Slack Distribution Technique for Multimode Distributed Real-Time Embedded Systems",* IEEE Transactions on Parallel and Distributed Systems, Vol. 16 No.7, July 2005.

[6] Sivakumar Viswanathan, Bharadwaj Veeravalli, and Thomas G. Robertazzi, *"Resource-Aware Distributed Scheduling Starategies for Large-Scale Computational Cluster/Grid Systems",* IEEE Transactions on parallel and Distributed Systems, Vol. 18 No. 10 October 2007.

[7] Regehr, Reid, Webbb, Parker, and Lepreau, *"Evolving Real-time Systems Using Hierarchical Scheduling and Concurrency Analysis," in 24th* IEEE Real-time Systems symposium, (Cancun, Mexico), Dec. 2003.

[8] Gabriel Ghinita, Panos Kalnis, and Yufei Tao, *"Anonymous Publication of Sensitive Transactional Data",* IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No.2, February 2011.

[9] Stavros Papadopoulos,liking wang, Yin Yang,Dimitris Papadias, and Panagiotis Karras, *"Authenticated Multistep Nearest Neighbor Search",* IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No.5, May 2011.

[10] Mohammad Alrifai, Peter Dolog, Wolf-Tilo Balke, and Woflgang Nejdl, *" Distributed Management of Concurrent Web Service Transactions",* IEEE Transactions on Services Computing, Vol. 2 No.4 October-December 2009.

[11] Sharadh Ramaswamy and Kenneth Rose, *"Adaptive Cluster Distance Bounding for High-Dimensional Indexing",* IEEE Transaction on Knowledge and Data Engineering, Vol. 23, No.6 June 2011.

[12] Yun Yang and Ke Chen, *"Temporal Data Clustering via Weighted Clustering Ensemble with Different Representations",* IEEE Transaction on knowledge and Data Engineering, Vol. 23, No.2 February 2*011.*

[13] Binoy Ravindran, *"Engineering Dynamic Real-Time Distributed systems : Architecture, System Description Language, and Middleware",* IEEE Transaction on Software Engineering, Vol. 28, No.1 January 2002.

[14] G. Goud, N Sharma, K. Ramamritham, and S. Malewar, *"Efficient Real-Time Support for automotive Applications: A Case study", Proceeding 12th* IEEE International Conference Embedded and Real-Time computing Systems and Applications, 2006.

[15] D. Chen and A.K. Mok, *"Scheduling Similarity-Constrained Real-Time Tasks",* Proceeding International Conference. Embedded Systems and Applications and International Conference VLSI, pp. 215-221, 2004.

[16] Victor C.S. Lee, Kwok-Wa Lam, Sang H. Son, Senior and Eddie Y.M. Chan, *"On Transaction Processing with Partial Validation and Timestamp Ordering in Mobile Broadcast Environments",* IEEE Transactions on Computers, Vol. 51, No.10 October 2002.