doi:10.1093/comjnl/bxh106

Capsule Reviews

FAIROUZ KAMAREDDINE

The Capsule Reviews are intended to provide a short succinct review of each paper in the issue, in order to bring the content to a wider readership. This issue's Capsule Reviews were compiled by Fairouz Kamareddine. Professor Kamareddine is an Associate Editor of *The Computer Journal* and is based in the School of Mathematical and Computer Sciences at Heriot-Watt University, Edinburgh, UK.

A Comprehensive Framework for Specifying Clairvoyance, Constraints and Periodicity in Real-Time Scheduling. K. SUBRAMANI

Usually, traditional scheduling models assume that the execution time of a job is constant in every instance of its invocation. In real-time scheduling models, this assumption is both inaccurate and dangerous. Moreover, unlike traditional scheduling models, real-time scheduling models are often constrained by complex time relationships between jobs. Under these requirements of a real-time scheduling system, a number of competing definitions exist for schedulability and it becomes difficult to choose the best one. This paper addresses the issue of schedulability through the degree of clairvoyance afforded to the dispatcher by the real-time application under consideration. In real-time systems, the dispatcher is concerned with strategies to assign jobs (from a job-set that he assesses to be feasible) on the time line. Depending on the application, there are three schedulability specifications: zero clairvoyance, partial clairvoyance and total clairvoyance, which cover cases where the dispatcher is unaware of the execution time of a job and where he knows the execution time of a job even before it commences. The scheduling framework proposed in this paper, the so-called execution time constraints (E–T–C) model, is shown to deal well with specifying intra-period constraints and is motivated through examples from real-time design. The E–T–C model is further extended into the E–T–C–P model that explicitly accommodates inter-period constraints needed in applications where the positioning of jobs within a scheduling window is affected by the positioning of jobs in the previous window and further affects the positioning of jobs in the next window.

DCOBE: Distributed Composite Object Based Environment. G. YILMAZ AND N. ERDOGAN

Distributed systems allow collaboration and co-operation between a number of interconnected computers and networks and enable the sharing of goals and data. In the recent design and implementation of computer systems, one sees a clear emphasis on collaborative information management and distribution. The main purpose of this paper is to introduce a new programming model for distributed systems. The proposed model, the so-called distributed composite objects (DCO) model, is based on 'composition' and 'replication'. Composition (or aggregation) allows aggregating multiple sub-objects into a single composite object and by doing so reduces the complexity. Replication allows for various copies of an object extending the object concept to the distributed environment. In DCO, sub-objects of a composite object are replicated on different address spaces to ensure availability and quick local access. No additional tasks are imposed on the application developer since the programing steps to create a DCO are similar to those required for a regular object. In order to conceal the implementation details of the DCO model behind the interface, the authors provide a transparent middleware software layer built on the Java Virtual Machine and known as the distributed composite object based environment (DCOBE). The DCOBE environment works on heterogeneous platforms. The creation and access of a DCO is illustrated by an example. Furthermore, in order to show the benefits of the DCO model, the paper uses a real-time collaborative writing system that allows two or more physically dispersed people work together on producing a book. Performance results which evaluate the DCOBE are also given.

Affinity-Based Routing in Zoned Mirrored Disks. A. THOMASIAN AND C. HAN

Mirrored disks store the same data at the two disks so that read requests can be processed by either disk, while write requests update both disks. In this way, mirrored disks are fault-tolerant, increase access bandwidth and decrease access time. Performance can be further improved by other means, in particular, by manipulating the routing of requests. This paper investigates a static affinity-based routing method which sends read requests destined to outer cylinders to the so-called outer disk, and those destined to inner cylinders to the so-called inner disk. Write requests are ignored in this paper since they enjoy no performance advantages in mirrored disks. Furthermore, the inner and outer disk cylinders are delineated by the so-called pivot point which helps in maintaining the balance between the loads of the outer and inner disks and in reducing the seek distance for some cylinders. Disks can be zoned (i.e. the number of sectors per track on outer cylinders is higher than that on inner cylinders) or not zoned (in this case, each track has the same capacity). In disks without zoning, the pivot point is

easy to find. It is in the middle disk cylinder since the inner and outer disks have the same capacity. For zoned disks, it is inadvisable to take a pivot point on the middle disk cylinder since this leads to an obvious imbalance. In fact, in zoned disks, the pivot point needs to be moved closer to the outer cylinder. In order to deal with this issue, this paper develops a method to evaluate the performance of zoned disks and proposes alternative pivot point selection criteria.

Biobjective Scheduling Algorithms for Execution Time–Reliability Trade-off in Heterogenous Computing Systems. A. DOĞAN AND F. ÖZGÜNER

A heterogeneous computing system consists of a collection of distributed high-performance machines interconnected by a high-speed network. Hence, the successful performance of a heterogeneous computing system depends on the performance of the machines and the network. Matching and scheduling algorithms minimize the execution time and the probability of failure. However, a trade-off exists between minimizing execution time and minimizing the probability of failure, and it may not be possible to simultaneously minimize both. Since these are two conflicting objectives, the paper is concerned with the so-called 'biobjective' scheduling problem. The authors rigorously formulate a biobjective scheduling problem where the first objective is the schedule length and the second one is the failure probability. The authors give a mathematical model that estimates the reliability of an application with precedence constraints. Unlike other models, this model is not restricted to tree network topologies. The biobjective scheduling problem is solved using two matching and scheduling algorithms: the 'biobjective dynamic level scheduling (BDLS) algorithm' and the 'biobjective genetic algorithm (BGA)'. The BDLS algorithm is obtained by modifying an existing scheduling algorithm (the DLS algorithm) and similar transformation steps can be followed for other existing algorithms. Furthermore, the BGA can be used as a benchmark algorithm to compare the performance of new algorithms. The paper carries out studies, which demonstrate that both algorithms enable the trade-off between execution time and failure probability.

Robust Data Compression: Variable Length Codes and Burst Errors. S. PERKINS AND D. H. SMITH

Variable length codes are used for data compression. However, although their use with enhanced error control can lead to benefits, their use with inadequate error control coding can lead to data loss. In order to avoid subsequent loss of data, one solution consists of encoding the data in small sections, together with a suitable synchronization scheme. However, synchronization may in some cases, lead to slippage where an error results in a difference in the number of symbols decoded before resynchronization and the actual number of data symbols that have been encoded. This may lead to a misinterpretation of the data. A form of synchronization that prevents slippage is called 'strong synchronization'. However, there is a problem with strong synchronization: it may be permanently lost during the recovery of synchronization since an entire cycle of code words may be jumped. An earlier paper of the authors in this journal (volume 47(3)) studied the various schemes used to avoid permanent loss of synchronization when very large sets of data are transmitted or sorted. The performance of these schemes was analyzed in the presence of high random error rates. This paper carries out a similar analysis for burst errors. A burst error is a set of consecutive positions in which some bit errors occur. This study both strengthens and complements the conclusions of the earlier mentioned paper and allows for a full analysis of the merits of the various schemes used to avoid loss of synchronization. Both data consistency checking and count consistency checking are introduced and it is shown that a count consistency check offers the best protection against long-term loss of strong synchronization.