doi:10.1093/comjnl/bxr114

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 it to a wider readership. The Capsule Reviews were compiled by Fairouz Kamareddine. Professor Kamareddine is an Associate Editor of *The Computer Journal* and is based in the Department of Mathematical and Computer Sciences at Heriot-Watt University, Edinburgh, UK.

Evaluation and Optimization of Kernel File Readaheads based on Markov Decision Models. CHENFENG XU, HONGSHENG XI AND FENGGUANG WU

The authors state the challenge to bridge the gap between disk drives and applications and focus on prefetching/readahead that is used to improve the file system performance, hiding I/O delays and improving disk utilization. Prefetching is usually integrated with caching. Furthermore, Markov prediction and Markov decision models have been used to identify complex access patterns, where prefetching actions are based on the predictions. However, the authors identify a lack of research on the analysis and optimization of kernel file fetching based on Markov decision models and propose the use of Markov decision models to describe the common readaheads at the kernel level, while explaining how to deal with the complexity of read handling and caching. A Markov decision process (MDP) is proposed to describe various readaheads integrated with the kernel read handling and caching and a performance gradient formula is derived which will be used to give an optimization algorithm. After an overview of related work, the common read handling and the readahead behavior are described at the kernel level. A policy is introduced so the kernel can decide whether to perform readahead and how many pages to prefetch. In his policy, the read sequence is given where the read handling process can be described using probability or stochastic models, but the readaheads are page based where the file cache is checked to make a decision on prefetching if the required page is missing. Hence, the page access process is given and the readahead state is introduced to trace the read sequence, the readahead actions and the selected rational readahead actions. To deal with memory issues, the cache state is introduced. All this sets the stage for the Markov decision model which the authors introduce along with performance metrics. Next, a performance gradient formula is introduced for the MDP and a prefetching policy optimization algorithm is given which provides a gradient optimization method based on simulations. The Markov decision method is then used to describe two readaheads in the Linux kernel: the legacy Linux readahead and the ondemand Linux readahead. The prefetching policies are then analyzed and optimized.

Dijkstra's Rallying Cry for Generalization: The Advent of the Recursive Procedure, Late 1950s–Early 1960s. EDGAR G. DAYLIGHT

The article focuses on early contributions of Dijkstra with respect to proposing and implementing the recursive procedure in the ALGOL60 programming language. At the time, it was debatable whether the recursive procedure should belong to a machine-independent programming language. This paper explains the tensions at the time between the pro- and anti-camps of the recursive procedure and its use or non-use by numerical analysts. An interesting question is asked in the paper with respect to the inclusion of the recursive procedure in ALGOL60 despite the fact that it was not needed by numerical analysts and that ALGOL60 was primarily designed for numerical computations. Important figures in this respect are classified as linguists (e.g. Dijkstra) or as those who are interested in machine efficiency (e.g. Stratchey). This is an interesting paper that touches on important concepts in the history of computation such as specialization versus generalization and optimization versus reliability.

Efficient Updates for OLAP Range Queries on Flash Memory. MITZI MCCARTHY AND ZHEN HI

Online analytical processing (OLAP) applications and data warehouses are important parts of decision-support systems. Summary structures have been proposed for answering OLAP queries; however, they are expensive to keep up to date. This paper outlines the importance of efficiently handling updates to data warehouses and OLAP data structures and proposes the use of flash memory in the form of solid-state drives (SSDs) rather than hard disk drives (HDDs) as the storage medium. However, random writes are much more expensive than random/sequential reads in SSDs and hence it is proposed to tap into the SSDs fast read performance while minimizing the

amount of writes in order to answer OLAP range queries much faster. The main idea is to trade more random reads to drastically reduce random writes. After an introduction to the NAND flash memory and a survey of related work, the methodology for storing and querying an OLAP cube in flash memory in the presence of updates is proposed. This is followed by various aspects of the experimental setup where three algorithms are compared. Four experiments are performed to compare these three algorithms. These experiments compare the algorithms for varying RAM size, varying prefix sum cube size, varying error bound, varying probability of update versus read queries, respectively.

Comparison of Allocation Schemes for Virtual Machines in Energy-Aware Server Farms. TIEN VAN DO

Server farms provide services in IT industry and give a quick and cost-efficient response to changes in workloads. However, there is a need to lower energy consumption. Virtualization can help save energy consumption. This paper considers dynamic requests for virtual machines to the server farm and proposes a number of actions to be applied to reduce energy consumption related to the operation of the server farm. A queuing model for server farms with physical servers and homogeneous demands is proposed first and is then extended to a model for heterogeneous physical servers and demands. Thereafter, the scheduling of homogeneous and heterogeneous virtual machines is given, and numerical results related to energy consumption under some priority assumptions are given.

Decimal CORDIC Rotation based on Selection by Rounding: Algorithm and Architecture. AMIR KAIVANI AND GHASSEM JABERIPUR

The paper states that the variants of the COordinate Rotation DIgital Computer (CORDIC) algorithm describe the displacement of a vector via micromovements on circular, hyperbolic or linear coordinates in rotation or vectoring models and that the determination of certain ingredients appear to be the most difficult tasks during iteration. The paper adds that solutions to rotation angles that can be easily determined lead to faster but more numerous iterations. The authors propose to handle rotation angles that are multiples of $\tan^{-1}(10^{-1})$ each within a single iteration and show that these multiples can be easily selected using the so-called 'selection buy rounding for radix-10'. First, the decimal CORDIC algorithm (in rotation mode for circular coordinates) is presented. Then, the selection of the microrotation factor by the rounding technique is presented. In order to reduce the latency of angle-selection, a retiming technique is defined. The authors then introduce computation and compensation techniques for the scaling factor and present the architecture. The proposed architecture is evaluated and is compared with previous work.

Integrating Information Theory in Agent-based Crowd Simulation Behavior Models. CAGATAY TURKAY, EMRE KOC AND SELIM BALCISOY

The paper states that believable behavior and appearance of a crowd improve a virtual environment's realism. As a step towards this direction, this paper proposes an analytical agent-based behavioral model that integrates global knowledge about crowd formation into local agent-based behavior control. Analytical maps are used to represent the activities of the crowd and keep track of the behavior of individuals within a specific space and time. These proposed so-called behavior maps are theoretically based on information theory and are automatically generated statistical analysis maps that record the spatial and temporal dynamics of the agents. It is claimed that this proposed model can extend any existing agent-based crowd simulator and, in particular, the authors extend the reciprocal velocity obstacles (RVO) multi-agent navigation system. The performance of the proposed model is discussed in an evacuation scenario simulation. After a review of the related literature, the authors introduce the analytical behavioral model that provides global knowledge on crowd's activities and enables the crowd simulator to incorporate agent-crowd interactions to modify agents' behavior. The crowd representation with behavior maps and the behavior map construction are given and followed by agent representation and a way to extend a crowd simulator with the proposed behavioral model. A number of tests are carried out to illustrate the performance of the crowd simulator and four test cases are reported in the paper.

Analysis of Performance-Impacting Factors on Checkpointing Frameworks: The CPPC Case Study. GABRIEL RODRIGUEZ, MARIA MARTIN, PATRICIA GONZALEZ AND JUAN TOURINO

Checkpointing is a widely used technique to obtain fault tolerance. New computing infrastructures present new constraints for checkpointing and raise the need for strategies for portable state recovery. This paper gives a thorough performance evaluation of ComPiler for Portable Checkpointing (CPPC) which is a checkpointing framework for message-passing applications with emphasis on portability. CPPC is an open-source checkpointing tool that inserts fault tolerance into long-running message-passing applications. The fundamental design aspects (such as portability, state file sizes, the coordination protocol and the CPPC compiler) of CPPC are first introduced and then followed by a runtime performance evaluation that evaluates the compiler and runs a number of experiments where 12 applications are selected for testing.

An Optimal Algorithm for Untangling Binary Trees via Rotations. JIA-JIE LIU, WILLIAM CHUNG-KUNG YEN AND YEN-JU CHEN

One way to measure the shape difference between two binary trees with the same number of nodes is to use rotation to

transform one tree into another such that the inorder sequences of the two trees are the same. The rotation distance dis(T, T') between any two n-node-rooted binary trees, where T and T' are the minimum number of left and right rotations needed to transform T into T'. It is not known whether dis(T, T') can be computed in polynomial time. An earlier work by Lucas proposed an $O(n^2)$-time algorithm for computing dis(T_0, T_f) with the minimum number of rotations, where T_0 is a degenerate tree (each node in T_0 has at most one child) and T_f is an angle tree (the right, respectively left, child of the root of T_f is not empty and every node in T_f having a non-empty right, respectively left, child is an ancestor of every node in T_f having a non-empty left, respectively right, child). This paper proposes an $O(n)$–time algorithm for computing dis(T_0, T_f) with the minimum number of rotations, where T_0 is a degenerate tree and T_f is an angle tree. After an introduction to Lucas's approach, the proposed $O(n)$–time algorithm is given.

Using Decision Tree Classifiers in Source Code Analysis to Recognize Algorithms: An Experiment with Sorting Algorithms. AHMAD TAHERKHANI

The aim of algorithm recognition (AR) is to abstract the purpose of the code and recognize the parts that can be improved. This paper aims to develop automatic AR methods. AR has a number of uses (e.g. automatically checking implemented algorithms and giving feedback, and source code optimization). This paper uses machine learning techniques and decision tree classifiers to recognize algorithms. The method is based on the static analysis of program code where algorithms are converted into characteristic vectors which are used to automatically build a decision tree classifier (using the C4.5 decision tree classifier algorithm) that can be used to recognize algorithms. After an overview of AR and related work, decision tree classifiers and the C4.5 algorithm are discussed. Thereafter, the proposed method for recognizing algorithms is given. The role of variables in the recognition and classification of algorithms is given high priority and a tool for detecting the roles of variables is presented. The proposed method is applied to five types of sorting algorithms (Quicksort, Mergesort, Insertion sort, Selection sort and Bubble sort) and the classification and recognition parts of these algorithms are described. It is shown how characteristic vectors and the classification tree are created. A number of experiments were carried out to evaluate the method for classification accuracy and contrasting the automatic and manual decision trees.

Matrix Implementation of Simultaneous Iterative Reconstruction Technique (SIRT) on GPUs. FRANCISCO VAZQUEZ, ESTER M. GARZON AND JOSE JESUS FERNANDEZ

Electron tomography (ET) makes it possible to visualize cellular environments at an unprecedented level of detail. Computational demands of iterative methods have prevented their extensive use in ET even though they are robust against noise and other conditions present in ET. In the last few years, graphics processing units (GPUs), using tricky features and low-level GPU programming, have revolutioned ET and have contributed to the acceleration of iterative methods. Recently, the authors have presented a fresh implementation of weighted backprojection (WBP) that exploits the matrix coefficients kept in memory and accelerates the tomographic reconstruction process. In this paper, the authors extend the matrix approach to iterative methods and evaluate it on different GPU platforms. After an introduction to ET and the the WBP construction method, the simultaneous iterative reconstruction technique (SIRT) is given and followed by the proposed matrix approach to iterative reconstruction. The matrix implementation algorithm of SIRT is then evaluated using several representative data sets on three different state-of-the art GPUs that were installed on different computers.

The L(2, 1)-labeling Problem on Oriented Regular Grids. TIZIANA CALAMONERI

The L(2, 1)-labeling problem refers to the frequency assignment problem of wireless networks where fixed antennas can both transmit and receive messages. This problem can be modeled as a coloring problem, where the aim is to minimize the maximum used color. This paper focuses on the oriented L(2, 1)-labeling problem on regular grids (squared, triangular and hexagonal grids), exactly determining the value of the maximum used color for each regular grid. After an introduction to the preliminaries needed for the paper, the author studies the oriented L(2, 1)-labeling problem (stressing the importance of the length of the longest dipath) first for squared grids, then for triangular grids and then for hexagonal grids. For these grids, the author has evaluated the smallest value of the length of the longest dipath such that a certain value of the maximum used color holds.

SE–Compression: A Generalization of Dictionary-based Compression. IONUT POPA

Inspired by the recent developments in grammar-based coding and its prospects on data compression theory, the author investigates the applications in data compression of the so-called synchronized extension (SE) systems. SE systems allow both simple concatenation and concatenation with overlapping in an unified manner. After an introduction to SE systems, the author introduces SE-compression which is almost the same as that of textual substitution but using an SE system instead of a dictionary and, implicitly, using the specific derivation rules when decompressing data. In SE-compression, overlapping blocks are allowed. Next, the author presents a method to use the SE systems combined with the classical LZW algorithm. The problems in the parsing algorithm in LZW (rigidity and the blocks resulting from parsing are not necessarily the most frequent blocks) are overcome by the proposed algorithm since it allows blocks to overlap. The

algorithm was implemented in C and tests were executed on OpenSolaris OS, Intel Pentium Dual Core 2.00 GHz and 3GB RAM to derive conclusive results and comparisons.

External Sorting on Flash Memory via Natural Page Run Generation. YANG LIU, ZHEN HE, YI-PING PHOEBE CHEN AND THI NGUYEN

This paper focuses on re-sorting partially sorted data (data that were initially sorted but becomes progressively unsorted due to updates). This has been studied in the literature but from the point of view of internal sorting (where all data fit in RAM). This paper concentrates on external sorting (of data residing in secondary storage). It is assumed that some mechanisms exist to inform when re-sorting is necessary. The traditional external merge sort algorithm is extended to take advantage of the partially sorted data and the unique characteristics of flash memory (which is fast at random reads but slow at writing). The key idea used is that the sorted run generation phase produces a lot of writes that can be minimized by finding disk pages that form a naturally occurring page run. After an introduction to flash memory and its characteristics, a detailed description of the tradition external merge sort is given since almost all external sorting algorithms are variants of this algorithm. Therafter, the authors explain the rationale for focusing on optimizing the write IO costs of run generation instead of the other IO costs associated with an external merge sort. The problem of finding naturally occurring page runs is formally described and then it is shown how it can be mapped into the well-known problem of finding the shortest path in a DAG. A heuristic approach is used to finding naturally occurring page runs of optimal size. This approach starts by loading as many pages as possible into the RAM buffer and then looking for non-overlapping pages among them. Six experiments are conducted using real and synthetic data sets where different sizes/results are varied (data size, record size, RAM size, etc).

Type Inference to Optimize a Hybrid Statically and Dynamically Typed Language. FRANCISCO ORTIN

The author argues for the benefits offered by dynamically typed programming languages, and states that these benefits have caused the recent addition of dynamic typing to some statically typed languages. He adds that the great flexibility of dynamic languages is counteracted by limitations due to the lack of static type checking that delays the detection of type errors (which in static typing happens at compile time, whereas in dynamic typing happens at run time) and slows runtime performance. Dynamic typing offers higher flexibility, whereas static typing implies better robustness and performance. In the literature, there have been arguments to use dynamic types even in the presence of advanced static type systems (static typing where possible, dynamic typing when needed). This paper presents the minimal core of StaDyn, a hybrid typing language that performs static type inference of both statically and dynamically typed references. The minimal core makes it easy to describe the type system and the erasure semantics. The compiler keeps performing type-checking even over dynamic references and the type information gathered at compile is used for both improving the runtime performance and the early type error detection of the programming language. In addition to the type system and the erasure semantics, the author carries out a runtime performance assessment to measure the runtime performance of the proposed method.