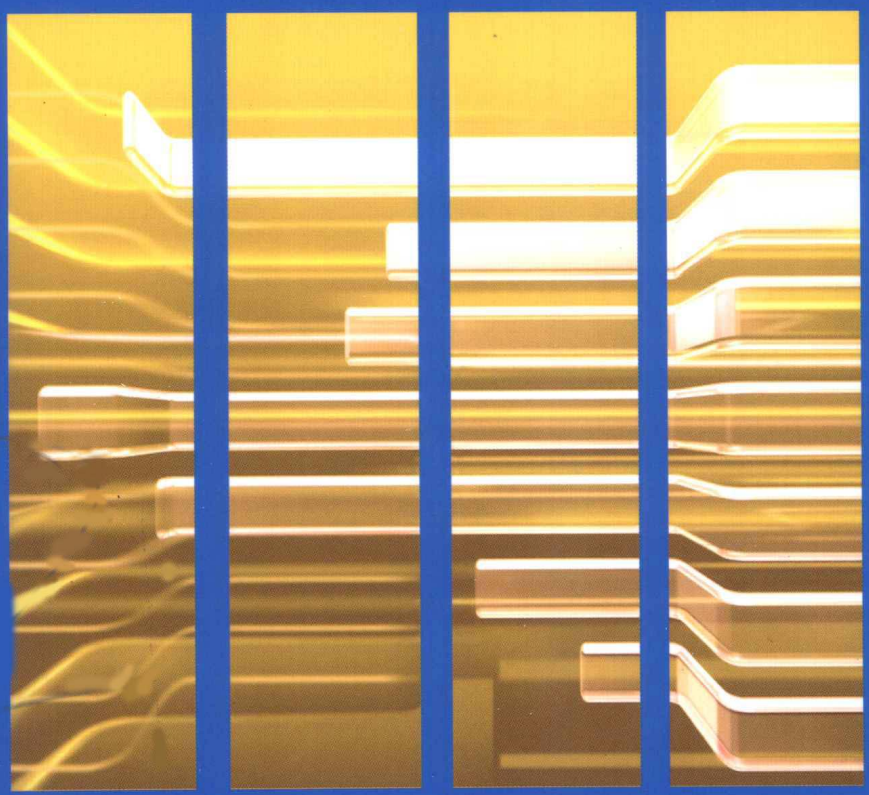
Proceedings of the Second International Symposium on Parallel Architectures, Algorithms and Programming

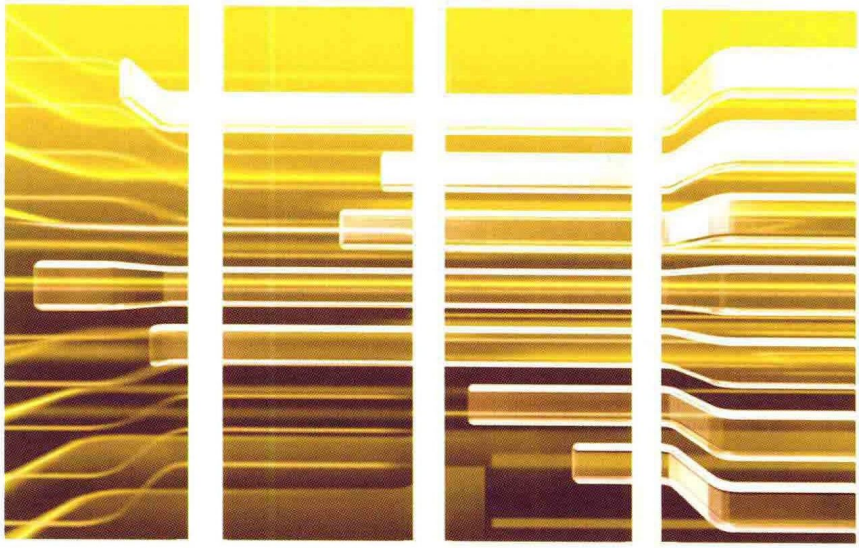
Edited by Ye Tian, Chen Zhong and Hong Shen



University of Science and Technology of China Press

Proceedings of the Second International Symposium on Parallel Architectures, Algorithms and Programming

Edited by Ye Tian, Chen Zhong and Hong Shen



University of Science and Technology of China Press

图书在版编目(CIP)数据

第二届国际并行体系结构、算法和程序设计研讨会会议论文集=Proceedings of the Second International Symposium on Parallel Architectures, Algorithms and Programming: 英文/田野，钟诚，沈鸿主编. —合肥：中国科学技术大学出版社，2009.12

ISBN 978-7-312-02538-9

I.第… II.①田… ②钟… ③沈… III.①并行计算机—计算机体系结构—国际学术会议—文集—英文②电子计算机—算法理论—国际学术会议—文集—英文③程序设计—国际学术会议—文集—英文 IV. TP338.6-53 TP301.6-53 TP311.1-53

中国版本图书馆 **CIP** 数据核字(2009)第 213187 号

Proceedings of the Second International Symposium on Parallel Architectures, Algorithms and Programming

edited by Ye Tian, Cheng Zhong and Hong Shen

96 Jinzhai Road

Hefei, Anhui

P.R.China

出版发行 中国科学技术大学出版社

地址 安徽省合肥市金寨路 96 号，邮编 230026

网址 http://press.ustc.edu.cn

印　　刷 安徽辉隆农资集团瑞隆印务有限公司

经　　销 全国新华书店

开　　本 710 mm × 1000 mm 1/16

印　　张 13.375

插　　页 1

字　　数 197 千

版　　次 2009 年 12 月第 1 版

印　　次 2009 年 12 月第 1 次印刷

定　　价 60.00 元

Conference Organization

Honorary Chairs

Guoliang Chen, Univ. Sci. & Tech. of China
Benjamin W. Wah, UIUC

General Chairs

Hong Shen, Univ. Sci. & Tech. of China
Baoshan Chen, Guangxi Univ.

Program Co-Chairs

Naijie Gu, Univ. Sci. & Tech. of China
Cheng Zhong, Guangxi Univ.

Technical Program Committee

Hamid Arabnia, Univ. of Georgia
Ling Chen, Yangzhou Univ.
Xuebin Chi, CNIC, CAS
Francis Chin, Univ. of Hong Kong
Qingshi Gao, Univ. Sci. & Tech. of China
Guangcan Guo, Univ. Sci. & Tech. of China
Yijie Han, Univ. of Missouri
Yanxiang He, Wuhan Univ.
Tao Jiang, Uni. of California - Riverside
Hai Jin, Huazhong Univ. of Sci. & Tech.
Guojie Li, ICT, CAS
Ming Li, Univ. of Waterloo
Minglu Li, Shanghai Jiao Tong Univ.
Xiaoming Li, Peking Univ.
Xuandong Li, Nanjing Univ.
Jianzhong Li, Harbin Institute of Tech.
Huimin Lin, IS, CAS
Zhiyong Liu, ICT, CAS
Zeyao Mo, IAPCM
Koji Nakano, Hiroshima Univ.
Lionel M. Ni, HKUST
Yi Pan, Georgia State Univ.
Yong Qi, Xi'an Jiaotong Univ.

Depei Qian, Beihang Univ.
Xubang Shen, CASC
Ninghui Sun, ICT, CAS
XianHe Sun, Illinois Institute of Tech.
Zhongxiu Sun, Nanjing Univ.
Feiyue Wang, IA, CAS
Baowen Xu, Nanjing Univ.
Zhiwei Xu, ICT, CAS
Xin Yao, Univ. of Birmingham
Wu Zhang, Shanghai Univ.
Xianchao Zhang, Dalian Univ. of Tech.
Xiaodong Zhang, Ohio State Univ.
Yunquan Zhang, IS, CAS
Weimin Zheng, Tsinghua Univ.
Zhengwei Zhou, Univ. Sci. & Tech. of China
Zhihua Zhou, Nanjing Univ.

Preface

Welcome to the second International Symposium on Parallel Architectures, Algorithms and Programming (PAAP 2009). The symposium is sponsored and organized by University of Science and Technology of China (USTC), Guangxi University, and China Computer Federation Technical Committee on High Performance Computing. The symposium is also supported by National Natural Science Foundation of China. PAAP'09 is an international forum for scientists, engineers, and practitioners to present their latest research ideas, progresses, and applications in all the areas of parallel and distributed computing with the focus on parallel algorithms, architectures and programming techniques.

University of Science and Technology of China (USTC) was founded by the Chinese Academy of Science (CAS) in 1958 in Beijing as a new type of national university. The university moved to Hefei, Anhui Province in 1970. Since its foundation, USTC has made distinguished achievements in talent fostering, scientific research and technology innovation. It has become an important base for top-quality talent training and high-level scientific research for the nation. According to the Ministry of Science and Technology, USTC is one of the best four universities in the science research performance in China. USTC ranks consistently among the best in the reviews of the Chinese top universities by the US journal "Science" and the French journal "Research".

The conference is hosted by Guangxi University at Nanning, Guangxi, China. Nanning is the capital of Guangxi Zhuang Autonomous Region of China, she is a city full of cultural distinctiveness, economic vitality, and an expanding openness to and involvement with the global community. She has received many awards including membership in "Top 50 Comprehensive Power Cities in China" and "Top Tourist Cities in China", in addition to being designated as a "China Hygiene Model City", the "Dubai International Award for the Best Practices to Improve Living Conditions" and the recipient of "Habitat Scroll of Honor Award" in 2007. The Annual Nanning International Folk Songs Festival in Autumn attracts widespread attention by combining the talents of musical headliners from the across the globe with a special blend of centuries-old folk song traditions, many "undiscovered" tourist attractions, and an ever-expanding economic trade. Since 2004, Nanning has hosted the annual China-ASEAN Expo sponsored by China and the ten ASEAN member states, at which China and ten of its Southeast Asian neighbors will offer a rich mixture of business opportunities, cultural experiences and tourism attractions.

Guangxi University was established in 1928. The university is in Nanning, the capital city of Guangxi Zhuang Autonomous Region. Located in charming subtropical scenery, the campus covers an area of 307 hectares with a building area of 745 000 square meters. The university has a library collection of over

2.02 million volumes, 3 national key disciplines, 6 national "211 Project" key construction discipline groups, 4 key laboratories of the ministerial level, 5 key laboratories of the provincial level, 21 provincial key disciplines, 1 demonstration base for teaching and research of modern agriculture technique, 46 research institutes or centers and 67 university and college laboratories.

We are very pleased to have seven distinguished researchers to present the keynote speeches at the symposium: "System Software for High Performance Computing " by Vice Chairman of CCF, Prof. Zheng Weimin from Tsinghua University; "Unified Change of System on Chip" by Academician of CAS, Prof. Shen Xubang from Xi'an Institute of Micro-Electronics; "High-performance Computers and Their Applications" by Academician of CAS, Prof. Chen Guoliang from USTC; "Green Computing" by Prof. Sun Yuzhong from Institute of Computing Technology, CAS; "Parallel Numerical Computing" by Prof. Sun Jiachang from Institute of Software, CAS; and "Interconnection Networks with Path Selection" by Prof. Fan Jianxi from Soochow University.

The three-day technical program contains keynote speech sessions, sessions for invited papers, and technical sessions. We would like to express our thanks to all the authors who have submitted their papers to PAAP'09. We also thank all the members of the Technical Program Committee, as well as all the external referees for their works and contributions in paper reviewing process. We would like to extend our special thanks to the members of the symposium organizing committee for their excellent work in organizing this successful event.

The proceedings contain qualified papers selected from submissions for presentation at PAAP'09. The publishing of the proceedings is supported by a number of projects supported by National Science Foundation of China under Grants No. 60533020, 60963001, 60563003 and 60772034.

Contents

A Hybrid Index Structure on Multi-core Cluster Architecture

Bai Long, GuangZhong Sun, Guoliang Chen

School of Computer Science and Technology
University of Science and Technology of China
Hefei, 230026, China
{blong, gzsun, glchen}@mail.ustc.edu.cn

Abstract. Since multi-core has been mainstream of processors, multi-core cluster architecture become more important for many server applications, including high-dimensional data processing. In this paper, we present a hybrid-index structure for high-dimensional data on multi-core clusters. To make full use of two-level parallelization of multi-core clusters, we design an index structure for high-dimensional data: HKD-tree(Hybrid K-Dimensional Tree). An HKD-tree is combined by KD-tree and LSH, which uses LSH in the leaf nodes of KD-tree. We parallelizes operations(tree construction and query processing) of HKD-tree. We evaluate the performance of HKD-tree with real image dataset. Due to the experiment results, HKD-tree is more efficiently for query processing on multi-core cluster architecture.

Key words: multi-core; parallelization; HKD-tree; cluster; high-dimensional data

1 Introduction

Over the past years, the development of CPU have been promoted by the development of processor technology and computer architecture. Since multi-core architectures were invented, it has been getting important effect in software development on cluster systems. It is generally believed that multi-core [1] architecture has good potential of performance and advantages to realize the following: Multi-core architecture can divide

a complex function of the processor chip into several cores to solve. So multi-core bring about the rapid upgrade of performance. Nowadays with the development of the size limit of the chip as well as the cost factors, the multi-core architecture have gradually become the mainstream of the computer.

Symmetric multiprocessor systems (SMPs) [2] are also becoming wides pread, both as compute servers and as platforms for high performance parallel computing. At present uses the SMP technology structure cluster node to become a tendency. In recent years, high-dimensional data index based similarity search has become one of the most important research area. One important problem among this area is how to efficiently find the similar feature vectors from the large amount of high-dimensional data spaces. Many applications such as image databases, medical databases, GIS (Geographic Information System) and CAD (Computer Aided Design)/CAM(Computer Aided Manage) applications require enhanced indexing for content based image retrieval. In this applications, how to indexing of high-dimensional data has become increasingly important [3].

Today, a lot of new index structures and algorithms have been proposed. There are several index structures for high-dimensional data spaces, such as KD-tree, LSH(locality sensitive hashing), R-tree, KDB-tree, X-tree, etc. As a general rule, these structures were used for the serial architecture of the hardware devices. However, with the constant development of multi-core technology, high-dimensional index structure parallelization has been put on the agenda of the index technical's development.

In our paper, we proposed an HKD-tree high-dimensional index structure, which is suitable for parallelization. Based on an HKD-tree index structure, we could make full use of two-level parallelization of multi-core clusters. We put the every computational nodes deal with the different subtask. We first do the query in the subtrees using OpenMP in every computational nodes. At the second, we merge the results which query in the subtrees using MPI. Therefore, every computational node could

parallel processing the subtask. An HKD-tree index structure is obviously suitable for this parallel architecture.

In section 2, we will offer a detailed description of the high-dimensional's current situation and existing problems. In section 3, we will describe an HKD-tree structure and its parallel algorithm specifically which we proposed. In section 4, we will aim at experiment's result to carry on the corresponding analysis to the structure. In section 5, we will make conclusions and remarks.

2 Current Situation and Existing Problems

During the last decade, with the development of multimedia technology, multimedia databases have become increasingly important in many application areas. An important research problem in the field of multimedia databases is the content based retrieval of similar multimedia objects such as images, text, and videos. However, like searching data in a relational database, a content based retrieval needs to query the similar objects as a basic requirement the database system. Most of the methods solving similarity search use a feature transformation which transforms important properties of the multimedia objects into high-dimensional points (feature vectors). Thus, the similarity search is transformed into a search of points in the feature space which are similar to query point in the high-dimensional data space. Query operation in the high-dimensional data spaces has therefore been a very important research area. To satisfy this requirement, some new index structures and algorithms have been invented. But they are not designed for the multi-core architecture. That is to say, they are not suitable for the multi-core cluster architecture. Next we come to specifically introduce several kinds of typical high-dimensional index structures.

2.1 KD-tree

The KD-tree is a data structure invented by Jon Bentley in 1979 [4]. Although it's too old, the KD-tree and its variants are still the most popular data structures used for searching in high-dimensional data spaces. In computer science, KD-tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional data space. KD-tree is a useful data structure for several applications. KD-tree is a special case of BSP(Binary Space Partitioning) trees. It uses a set of k keys to partition k-dimensional space.

If you want to construct a KD-tree, and you have a set of n points in a k-dimensional space, the KD-tree is constructed recursively as follows: At the first, we find a median of the value of the ith coordinates of the points (initially, $i = 1$). That is, a value M is computed, so that at least 50% of the points have their ith coordinate equal or greater than M, while at least 50% of the points have their ith coordinate equal or less than M. The value of x is stored, and the set P is partitioned into P_L and P_R, where P_L contains only the points with their ith coordinate equal or less than M, and $|P_R| = |P_L| \pm 1$. When all the points have their own position, the recursion stops. Its construction algorithm and query algorithm is as follows: Algorithm1 and Algorithm2

2.2 LSH

LSH(locality sensitive hashing) was first introduced by Indyk and Motwani [5]. LSH function families have the property that objects that are close to each other have a higher probability of colliding than objects that are far apart. Specifically, let S be the domain of objects, and D be the distance measure between objects.

Definition. A function family $H = \{h : S \to U\}$ is called (r, cr, p_1, p_2)-sensitive for D if for any $q, p \in S$

- If $D(q, p) \leq r$ then $P_{rH}[h(q) = h(p)] \geq p_1$,

```
KD-Tree Construction
    if A=Null then
      root:=P
    else if A is the Lchild of X then
      X.Lchild:=P
    else
      X.Rchild:=P
    else if P[i]<A[i] then
      KD-Tree Insert(A.Lchild,P,A)
    else
      KD-Tree Insert(A.Rchild,P,A)
    end if
```

Algorithm 1: KD-Tree Construction

```
KD-Tree Query Processing
    if A=Null then
      return Null
    else if A=P then
      return A
    else if P[i]<A[i] then
      KD-Tree Search(A.Lchild,P)
    else
      KD-Tree Search(A.Rchild,P)
    end if
```

Algorithm 2: KD-Tree Query Processing

- If $D(q,p) > cr$then $P_{rH}[h(q) = h(p)] \leq p_2$.

To use LSH for approximate nearest neighbor search, we set $c > 1$ and $p_1 > p_2$. With these choices, nearby objects (those within distance r) have a greater chance ($p_1 vs. p_2$) of being hashed to the same value than objects that are far apart (those at a distance greater than cr away). Different LSH families can be used for different distance functions D. Families for Jaccard measure, Hamming distance, l_1 and l_2 are known [5]. Datar [6] et al. have proposed LSH families for l_p norms, based on p-stable distributions [7]. Here, each hash function is defined as:

$$h_{a,b}(v) = \lfloor \frac{a \cdot v + b}{W} \rfloor$$

where a is a d-dimensional random vector with entries chosen independently from a p-stable distribution and b is a real number chosen uniformly from the range [0,W]. Each hash function $h_{a,b} : R^d \rightarrow Z$ maps a d-dimensional vector v onto the set of integers. The p-stable distribution used in this work is the Gaussian distribution, which is works for the Euclidean distance.

LSH algorithm as follows: Algorithm3 and Algorithm4

LSH Construction

Make the Point Dataset X into binary string of Hamming Space

Choose k functions $h_1, h_2, ...h_k$ uniformly at random(with replacement) from H. For any $p \in X$, place p in the bucket with label g(p)= $(h_1(p), h_2(p), ...h_k(p))$. Observe that if each h_i outputs one "digit",each bucket has a k-digit label.

Independently perform step(1) l times to construct l separate hash tables, with hash functions $g_1, g_2, ...g_l$.

Algorithm 3: LSH Construction

LSH Query Processing

1: For query q, make use of hash function from the Algorithm3, extract $g_i(q), 1 \leq i \leq k$, the selected hash tables entry

2: On the order of retrieve these tables, we can got the query results.

Algorithm 4: LSH Query Processing

2.3 Others

Other high-dimensional index structures include: KDB-Tree [8], R-Tree [9],S-Tree [10], SH-Tree [11],and so on. Based on the above analysis, we can see that the serial structure of the above-mentioned cases have substantial efficiency. But when they applied to such a parallel system of cluster structures, it seems inappropriate. Therefore, in these high-dimension based on the index structure, we give an overview of efficient parallel search algorithm and the hybrid-index structure for high-dimensional data under the cluster architecture.

3 Solutions

In this section, we introduce the hybrid kd-tree(HKD-tree). We discuss how to construct a HKD-tree and some operations in the hybrid tree. We also discuss the node splitting algorithms and show how they parallel expected search performance. We describe the tree operations and conclude with a discussion on where the hybrid tree fits parallelization under the cluster architecture. Because most of search including the on-line and off-line model, we from user's angle mainly discuss the on-line performance. That means, we primarily concern the query performance.(Our CBIR: http://search.ustc.edu.cn/cbir/)

3.1 HKD-tree Sructure

We first discuss the hybrid-index structure for high-dimensional data under the cluster architecture. Similar to KD-tree, HKD-tree is also a

space-partitioning data structure for organizing points in a k-dimensional space. We have proposed the hybrid-index structure under the cluster architecture to solve the high-dimensional index problem. HKD-tree means Hybrid KD-tree. In this hybrid-index structure, we have KD-tree and LSH mixed. For the high-dimensional data,we use KD-tree in construction index. When we do the query in it,we can use LSH to solve the Nearest Neighbor Search.

First, we describe the "space partitioning strategy" in a HKD-tree i.e. how to partition the space into two subspaces when a node splits. The first issue is the number of dimensions used to partition the node. A HKD-tree always splits a node using a single dimension. A HKD-tree is a binary tree in which every node is a k-dimensional point. Every non-leaf node generates a splitting hyperplane that divides the space into two subspaces. Points left to the hyperplane represent the left subtree of that node and the points right to the hyperplane by the right subtree. The hyperplane direction is chosen in the following way: every node split to subtrees is associated with one of the k-dimensions, such that the hyperplane is perpendicular to that dimension vector. When we do split to the parallel start layer, we set the subtree's node into the hash table by hash function. So, for example, if for a split the "x" axis is chosen, all points in the subtree with a smaller "x" value than the node will put in the left subtree and all points with larger "x" value will be in the right subtree. At the same time, we collect all the data points add into point dataset X. HKD-tree algorithm as follows: algorithm5 and algorithm6, and an HKD-tree structure is as follows Figure 1:

It is clear from the above discussion that a HKD-tree structure is more suitable for parallel. Compared to other tree structure, HKD-tree make the following improvements: The first change is in the representation. As in other tree index techniques, they often use single index structure. Such as KD-tree, R-tree, B-tree, LSH. etc. Therefore, their advantages and weaknesses are highlighted. But in HKD-tree, we mixed KD-tree and

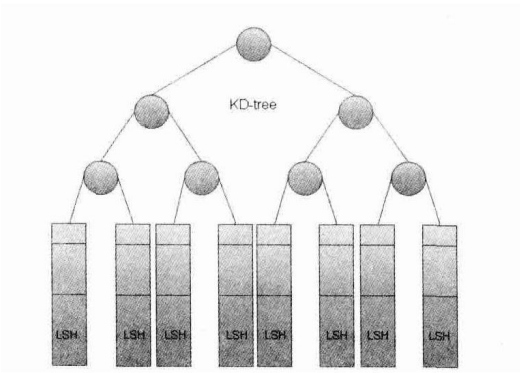


Fig. 1. HKD-Tree Structure

HKD-Tree Construction

if A=Null **then**
  root:=P
else if A is the Lchild of X **then**
  X.Lchild:=P
else
  X.Rchild:=P
else if P[i]<A[i] **then**
  HKD-Tree Insert(A.Lchild,P,A)
else
  HKD-Tree Insert(A.Rchild,P,A)
else if our cpu have n cores and our HKD-tree have m subtrees in $\log 2^{m+1}$ layer. **then**
  when m = n,we mapped these subtrees nodes into point dataset X
  point dataset $X = \{X_1, X_2, ...X_n\}$
  That is mean, nodes of subtree m_1 mapped into point dataset X_1
end if

Algorithm 5: HKD-Tree Construction