Efficient Parallel Permutation-Based Range-Join Algorithms on Mesh-Connected Computers

Shao Dong Chen  Hong Shen  Rodney Topor

School of Computing and Information Technology
Griffith University
Nathan, Queensland 4111, AUSTRALIA
Email: {schen,hong,rwt}@cit.gu.edu.au

Abstract. This paper proposes three efficient parallel algorithms for computing the range-join of two relations on two-dimensional \( n \times m \) mesh-connected computers, where \( n \) and \( m \) are the numbers of the rows and columns respectively. After sorting all subsets of both relations, all proposed algorithms permute all sorted subsets of one relation to each processor in the computers, where they are joined with the subset of the other relation at that processor by using a sequential sort-merge range-join algorithm. The Min-Storage-Shifting and Min-Movement-Shifting algorithms permute the data on a mesh alternatively in the row and column directions, and Hamiltonian-cycle algorithm permutes the data along a Hamiltonian cycle of the mesh. The analysis shows that the Hamiltonian-cycle algorithm requires fewer local join operations but more data movements than other two algorithms and that the Min-Movement-Shifting algorithm requires fewer local join operations and data movements but more storage than the Min-Storage-Shifting algorithm.

Keywords: Relational Databases; Parallel Query Optimization; Range-join; Parallel Algorithms; Analytic Cost Models; Mesh-connected Multiple Computers.

1 Introduction

This paper presents and analyzes three new efficient parallel algorithms on two-dimensional mesh-connected computers for computing the range-join operations, which are the generalization of the conventional equi-join and band-join operations [5]. For two given constants \( e_1 \) and \( e_2 \) with \( 0 \leq e_1 \leq e_2 \), the range-join of two relations \( R \) and \( S \) on attribute \( A \) from \( R \) and \( B \) from \( S \), denoted by \( R \bowtie_{e_1}^e S \), is the relation \( T \) obtained by concatenating all tuples \( r \) in \( R \) and \( s \) in \( S \) such that \( e_1 \leq |r.A - s.B| \leq e_2 \) [10]. Range-join operation is an important operation in relational database systems and appears frequently in practice, especially in the queries requiring joins over continuous real world domains such as time and distance. Moreover, since range-join operation is the generalization of the conventional equi-join and band-join operations, the range-join algorithms can be
used to compute equi-join and band-join operations directly without any loss of efficiency.

As the join condition of range-joins involves range comparisons rather than equalities, hash-based join algorithms are unsuitable for range-join operations [5]. In contrast, permutation-based join algorithms, which are the parallel versions of sequential nested-loop join algorithms, have been shown to be effective for computing range-joins on hypercube computers [10]. In general, with the assumption that each relation is distributed evenly across all processors initially, permutation-based algorithms sort all subsets of both relations R and S, then permute each subset of S to every processor in turn, where it is joined with the local subset of R at that processor. The local range-join operation in each processor for two sorted subsets is implemented by a sequential sort-merge algorithm SSMRJ.

Obviously, the major problem for developing permutation-based range-join algorithms on a mesh-connected computer is how to permute data efficiently. In this paper, we develop two permutation methods which result in three different algorithms: Min-Storage-Shifting and Min-Movement-Shifting join algorithms which permute data in the row and column directions alternatively, and Hamiltonian-cycle join algorithm which permutes data along a Hamiltonian cycle of the mesh. We analyze and compare these algorithms in terms of running time and storage requirement. The Hamiltonian-cycle algorithm requires less storage and local join operations but more data movements than the other two algorithms; the Min-Movement-Shifting algorithm requires less local join operations and data movements but more storage than the Min-Storage-Shifting algorithm.

The remainder of this paper is organized as follows. In Section 2, we introduce the mesh computers and describe the permutation-based join algorithms in general. Our sequential sort-merge algorithm SSMRJ is presented and analyzed briefly in Section 3. The three mesh parallel range-join algorithms, which use this sequential algorithm in each individual processor, are presented and analyzed in Sections 4, 5 and 6 respectively. Section 7 concludes the paper by comparing the performance of these three parallel algorithms.

2 Preliminaries

A mesh-connected computer ("mesh" for short) is an SIMD parallel computer with mn processors, each having its own local memory and disk, without shared memory or disk. These processors are connected in an n \times m two-dimensional grid. The processor in row i and column j is denoted by P_{i,j}, where 1 \leq i \leq n and 1 \leq j \leq m. Each processor can transfer data from its local memory to one of its neighbors’ local memory; concurrent data movements are allowed only when they are all in the same direction. There is no wraparound edge between the boundary processors and hence it presents more challenge tasks for designing algorithms on meshes than on toruses which have such wraparound edges.

Due to its flexibility and cost efficiency, the mesh is an important parallel computer system which supports many parallel algorithms [9] and has been used