HPPC Project

Research Paper Notes

McKenzie Wierda

11/24/19

**Paper 1**

**Title**: A Parallel Implementation of the K Nearest Neighbours Classifier in Three

Levels: Threads, MPI Processes and the Grid

**Authors**: G. Aparı́cio, I. Blanquer, V. Hernández

Overview

1. their work focuses on the problem of data mining and classification of large amounts of data using K nearest neighbors classifier
2. they solved the computing demand with a parallel computing implementation designed to work in Grid environments of multiprocessor computer farms
3. They combined parallel techniques (intra-node, inter-node, and inter-organization) to complete a 1-month CPU job in a few hours
4. Technologies used are EGEE Grid Computing Infrastructure with Large Hadron Collider Computing Grid middleware, and MPI and POSIX threads

Intro

1. Data mining is a concept that groups different techniques of data analysis and model extraction
   1. used primarily to extract hidden, predictive information from large databases
2. Their focus is classification
   1. Automatic choice of the category in which a piece of information will more likely fall into, which KNN is well-suited for
3. Focus on speeding up performance; analysis of accuracy and goodness of predictions are not their purpose

K nearest neighbors

1. Appropriate when dealing with a large set of labeled registers or instances and a small set of non-labeled registers to be classified
2. The K parameter is very important and must be chosen experimentally
3. Cross Validation Test is the most popular technique used
   1. Starting with a set of labeled registers, break them into *B* blocks
   2. One block used as a test set, and the others as the training set
   3. The KNN classifier is applied to any register in the test block and compares the label assigned with the one previously defined
   4. If the labels differ, an error is recorded, and the consolidation of all the errors obtained for the whole test block is the validation error
   5. Repeat by changing which block is the test block
   6. The sum of all errors is an approximation to the real error of the KNN classifier with chosen K parameter

Evaluation of Error

1. For a database of 1 million records and 20 fields per record, the eval of error takes 18 CPU hours to complete on a state-of-the-art computer
2. A complete optimization process of 10 different K values would take over 7 CPU days
   1. unmanageable in a production environment

Parallelism

1. The process is intrinsically parallel, with two clear levels
   1. Each evaluation of the error using different values of K is totally independent, since it consists of computing the whole classification and cross-validation process for each value of *K*, using the same input
   2. The process consists of computing the *K* minimal distances to all registers and selecting a block to act as the test set
   3. This process is repeated, selecting different blocks of the database as training sets to cover the whole input database
   4. Each validation using a different block is independent
   5. Computation cost = cN^2
      1. c = number of flops for computing a single distance between two registers
      2. N = number of registers in the database
2. Grid Computing
   1. Coarsest granularity
   2. Deals with concurrent usage of different computing resources in different administrative domains, similar to large-scale batch queue
   3. Inter-resource communication not usually available
   4. Data access performed through job submission process and shared repositories
3. Message-Passing Parallel Computing (MPI)
   1. efficient in medium-grain problems where communication costs are lower than computing costs
   2. Jobs are fairly symmetrical and run on homogeneous nodes connected through fast network
4. Shared-Memory Parallel Computing
   1. Finest-grain parallelism
   2. Applicable in coupled and homogeneous environments
   3. Different threads concurrently execute a common program on different fragments of data
   4. Scaling factor of those systems is low, due to hardware constraints and speed-ups are good
5. They used all three to achieve maximum performance
   1. Shared-memory only gets small speed-ups b/c of limited processors
   2. Combined with MPI gives better speed-up, where computing farms can reach many tens of processor nodes without performance loss
   3. But with Grid, they can coordinate the usage of several computing farms at once
      1. Grid is also an efficient way to organize and manage resources

Implementation (three components)

1. Parallel KNN module
   1. implements KNN and cross-validation algorithm using MPI and POSIX threads
   2. Autonomous executable that takes as input the reference to the labeled registers file name, the reference to the file that contains the registers we want to label, the K value, and produces a different output file depending on their demands
   3. Ensures the fields of each register are considered with the same weight, and evenly distributes them among the processors
      1. and inside the processors, evenly among the threads
2. Grid Scripts
   1. Implement the selection of rightmost computing resources, job description file, start-up script for the executable, job submission, monitoring, and output retrieval
3. Java Interface
   1. User-friendly interface to select the data and parameters for Grid jobs and retrieve their output
4. Three-layer parallelism scheme
   1. The root of the tree is handled by Grid, which submits K different Grid jobs with different K values
   2. MPI parallel processes are executed, equal to the number of cross-validation blocks
   3. MPI processes are split into threads to execute KNN

Result

1. Linear speed-up was obtained with grid technology
2. MPI parallelization provided a 9.5 factor gain with 10 processor nodes
3. 4 threads per node gave an additional 1.5 factor gain in speed
   1. Each of their nodes had 2 hyper-threading processors
4. Total gain using MPI and thread technologies was over 15

**Paper 2**

**Title:** Parallel algorithms for nearest neighbor search; Problems in high dimensions

**Authors:** Bo Xiao and George Biros

Overview

1. Present algorithms and a library built on top of MPI and OpenMP that enable nearest neighbor searches to hundreds of thousands of cores for arbitrary dimensional datasets

Intro

1. KNN = *all nearest neighbors* problem
   1. Solving is easy by direct search in O(m\*n) work
      1. For each query point, all distances to the reference points are evaluated, followed by a k-selection problem in a list of N numbers
   2. This demands O(N^2) work, which is prohibitive when N is large
      1. Note: Even spatial data structures fall apart in higher dimensions, moving from O(N\*log(N)) complexity back to O(N^2)
2. Motivation
   1. KNN is a fundamental problem and a building block for higher-level algorithms
   2. But, little work has been done to scale it to high-performance, parallel platforms

Approach

1. Propose two parallel algorithms for direct calculation of KNN problem
   1. first one prioritizes computing time over memory by replicating the query and reference points while minimizing synchronization costs
      1. useful for absolute wall-clock performance
      2. but not memory optimal
   2. Second algorithm uses a cyclic iteration that is memory optimal but uses more communication

Methods and results

1. Single-node distance and k-nearest neighbor kernels
   1. Single-node, multi-threaded kernels for distance calculations and KNN searches on locally stored points
   2. It calls a distance routine and sorts the squared distances of each query point in ascending order, while tracking the index of the reference point for each distance
   3. Rather than sorting each row of the input, they kept a minimum heap for the first k nearest points
      1. Gives a total complexity of O(Nlog(k))
   4. Ran into problems where multi-threaded performance when multiplying a tall, skinny matrix by a relatively small matrix
2. Brute-force Direct
   1. Two dimensional partitioning
      1. Divided the reference and query points into r and q pieces, and distributed across r \* q nodes
      2. Each process computes a matrix containing the distance between each (r,q) pair, and sorts each row of the matrix to select the k minimum distance
   2. Cyclic partitioning
      1. Does not replicate data across processes, which let them solve KNN for significantly larger problem sizes on a single node
      2. But costs additional CPU time
      3. r and q partitioned into ‘p’, nearly-equal sized partitions distributed among processes
      4. Functions by running the local KNN kernel and merging the results with the current k minimum distances for each point
         1. At each communication step (p total), the partitions of r are processed in a ring of processes
      5. Time complexity is determined by the time spent in local direct KNN calculation
         1. ??? O( mnd/p + m + n + (mn/p)(log(n/p)))