Hybrid MPI and GPU Approach to Efficiently Solving Large kNN Problems

Libin Sun, Cyrus Stoller, Tia Newhall
Swarthmore College, Department of Computer Science

K-Nearest Neighbors (kNN)
An instance based algorithm used to classify queries using information from the k-nearest neighbors from a set of pre-classified reference points (training set).

Examples of uses of kNN
- Computer vision - face and fingerprint recognition
- Information retrieval
- Music genre classification
- High Dimensional Feature Space
  - • Documents of the World Wide Web
  - • GIS Data
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases
  - • GIS Data
  - • Documents of the World Wide Web
  - • Scanning CCTV footage for facial recognition
  - • Scientific Databases

High Dimensional Feature Space
- Sorting calculated distances to determine the k-nearest neighbors

Our Contribution
1. Hybrid parallelization of this algorithm across a cluster and on individual nodes using a GPU.
2. Solves bigger problems than existing CUDA approaches
3. Faster than existing MPI approaches
4. Methodology can be applied to other problems

kNN algorithm (2D example)
1. Calculate distance to all reference points
2. Sort the distances
3. Take a vote of the k-nearest neighbors (k=5 in the example to the left) to determine the color for “?”

Using MPI and CUDA for scalable, fast kNN solution
- MPI: distribute reference data and computation among worker nodes
- CUDA: compute and sort distances to retrieve k nearest neighbors for all of the queries on its subset of reference data

Two Layers of Parallelization
- MPI Layer
  - Master Node (iRank = 0)
- Worker Node (iRank = 1, 2, ..., n)

- CUDA Layer
  - kNN on each node:
    - while (not done)
      - CPU sends next chunk of data to GPU
      - if (master process)
        - compute distances
    - else
      - sort distances
      - CPU receives results from GPU
      - keep results in a priority queue
      - write known k-nearest neighbors for each query to file

Merging Final Results
- MPI reduction merge:
  - Determine number of levels in reduction
  - At each level, create temporary result files
  - Write known k-nearest neighbors from previous level of result files

Making the merge step faster:
- Reduction merge to take advantage of parallelism
- Use a priority queue in each merge phase to reduce the amount of file I/O for finding the top-k results

Results from TeraGrid
Conducted these on Lincoln from NCSA
96 Nodes, Intel 64-bit Linux Nodes, 2 NVIDIA Tesla S1070 Accelerator Units per node

Speedup vs. # Nodes vs. problem size
- • speedup close to linear for large problems
- • speedup limited by merge time in smaller problems

Total Execution Times
Time in seconds with varying num reference points and num nodes. For all experiments, k=20, num query points=60,000

kNN and Merge Time Breakdown
- • up to 80+ times faster than using CUDA alone
- • 17 minutes to solve kNN of 60,000 queries against 160 million reference points (76GB of floats)

References
2. National Center for Supercomputing Application. Intel 64 tesla linux cluster. www.ncsa.illinois.edu