RISE:2019
Abstract ID#: 2646
Category: Computer and Information Science
Status: Graduate
Northeastern University

Pi-Tiles: A Distributed Matrix-multiplication Approach over Tiled Data using Raspberry Pis

Kaustubh Shivdikar, Ritvik Rao, Rubens Lacouture, Gabriel Peter, Benjamin Knower, Andrew Jacob, David Kaeli

Opportunity

Abstract

- Machine learning allows computers to analyze data and complete tasks based solely on observations of patterns
- Effective deep learning relies on the ability of parallel processors to transform large amounts of data in the shortest possible time
- This project aims to demonstrate a parallel version of the tiled/blocked matrix multiplication method

Introduction

- MPI (Message Passing Interface) is an open-source software for parallel computation [1], which can allow one to take advantage of cheap, single-core processing units like Raspberry Pi's
- The previously slow nature of microprocessors now has the ability to "divide and conquer" problems or calculations at a very significant increase in speed
- Such applications for a multi-core setup are diverse and numerous; however, the scalability and high calculation count of matrix multiplication made it a suitable task to execute over a Raspberry Pi cluster using MPI

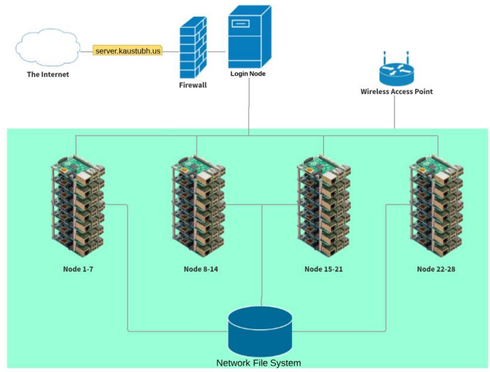
Goal

- Matrix multiplication is the building block of various scientific and engineering applications
- Computing matrix multiplication is a computationally demanding task
- We aim to reduce the time for computing matrix multiplication by using a tiling/blocking method to improve temporal locality, in addition to separating the task among various compute nodes
- We achieve this by partitioning the matrices into submatrices (blocks) and distributing them to the compute nodes for them to compute the resulting matrix in parallel

Approach

Method

- Two matrices, a matrix A and a matrix B, are multiplied together to produce a resultant matrix C
- For data collection, all matrices are square matrices and have dimension sizes equal to powers of 2
- Matrices A and B are divided into "tiles"/"blocks" that represent a subsection or each array
- The matrix multiplication is divided into "tasks" that multiply one tile from A to a corresponding tile from B
- Each task is allocated to each of the Raspberry Pis, and the tiles for each task are given to each node using the MPI_Send and MPI_Recv functions [2]
- Once each node completes a task, the result is delivered back to the master node
- The master node then adds the result to the corresponding location in matrix C
- Execution time of multiplication is measured with MPI_Wtime

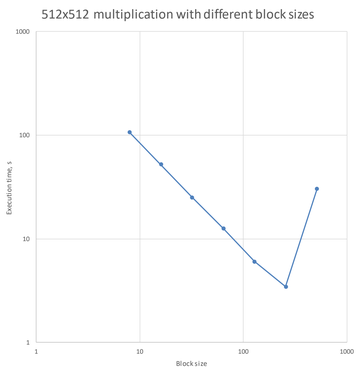


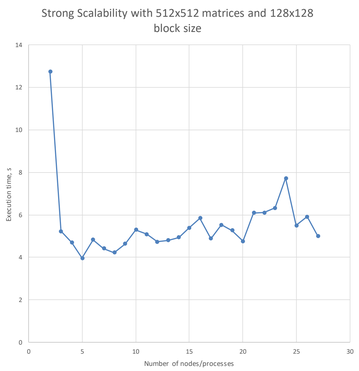
Implementation

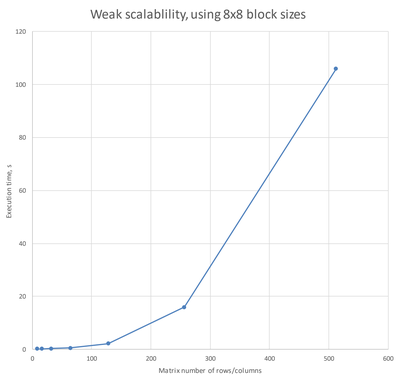
- Hardware: Cluster of 28 Raspberry Pi 3 B+ computers, each with quad-core ARM-v8 Cortex-A53 CPU, with a clock speed of 1.4 GHz and 1 GB of RAM
- Software: Message Passing Interface, also known as MPI, implemented in C

Data/Results

- Matrix and tile sizes used for measurements: 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, 256x256, 512x512
- Changing block size: Fastest operation for multiplying a 512x512 matrix took 3.45 seconds on average, using 256x256 block size
- Strong scalability: When multiplying a 512x512 matrix with 128x128 block size, multiplication time held steady as the number of processes increased
- Weak scalability: When size of tasks kept constant (multiplying 8x8 tiles), time of execution increased exponentially with matrix size, but also executed within 2 minutes in the worst case
- Speedup compared to non-tiled multiplication: 37 seconds avg. vs 3.45 seconds best case for 512x512 matrix (about 10x speedup)
- Speedup, all nodes vs. single node tiled multiplication: 12.73 seconds vs. 6 seconds average for 512x512 matrix and 128x128 block size (about 2x speedup)







Impact

- Addresses the problem of the limit on the speed at which matrices can be multiplied on a large scale
- On a single processor, there is a limit on the ability to improve efficiency and multiply matrices faster.
- When using multiple processors, it is possible to circumvent the shortcomings of a single processor by distributing the computations across each processor
- Each calculation that must be done in order to fully multiply the matrices can be split up into smaller tasks by the master node, each of which can be given to one of the other nodes to deal with
- Because matrix multiplication is so computationally expensive, it is one of the parts of deep learning that takes the longest to perform. Speeding it up in this way makes deep learning easier and more practical, since it will take so much less time.
- Applications that will benefit from tiled matrix multiplication include image processing, language processing, and big data analysis [3]

References and Acknowledgements:

[1] MPITutorial.com, a comprehensive MPI tutorial resource created and maintained by Wes Kendall
[2] MPICH.org, featuring a full documentation from MPI
[3] Schmidt, Uwe; Roth, Stefan. *Shrinkage Fields for Effective Image Restoration*. Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference