ECE574 Final Project-Calculate PI Benchmark

He Qihao
ECE574
May 10, 2017

1. Introduction:

What your project is and what the goal was.
My project is to study the parallel code implement in Message Passing Interface(MPI). My goal is to code a MPI benchmark that runs on distributed systems and shared memory systems.

2. Related work.

List and properly cite at least three related projects that did similar work (academic papers would be best, but you can also cite websites and such). Describe how your work differs from the previous work.

Parallel computing applies in calculation.

Ordinary Differential Equation(ODE)

Some ODE is data dependent, make it not suitable for parallel computing.

Related work: Parallel methods for the numerical integration of ordinary differential equations [1]

Optimization problem

Solve initial condition boundary optimization problem. Steepest descent algorithm depend data that requires the current data point to the next point for the gradient, then process for the line search.

Related work: Parallel global optimization with the particle swarm algorithm [2]

Matrix multiplication

Easy to implement, easy to understand, but the topic is too common, does not have distinguish feature that is difficult

Prime number finding

Prime number finding in a certain range. Using certain mathematical formulas to find prime number, or using divide by integers and have remainder equals 0 to filter out the non-prime number.

Related work: 2 Fast Parallel Prime Number Sieves [3]

Integral of continuous function.

Suitable for parallel computing. The higher the sample frequency the better the precision.

Double precision floating point variable data type.

Related work: Highly Parallel, High-Precision Numerical Integration [4]

3. Experiment setup

hardware

(a) Describe the computing hardware that you run on. Say if it is shared memory, a distributed system, or something else. List the CPU (architecture, type, speed), RAM, and network.

Share memory system:

VW-Haswell machine,

CPU: x86_64, 32 cores, model name: Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz, RAM: 82366408 kB.

Ethernet connected.

Distributed system:

VMW Research Group Raspberry Pi Cluster:

CPU: Pi2B -A 900MHz quad-core ARM Cortex-A7 CPU,1GB RAM

24-node pi 2 cluster,96 cores with 24GB of RAM.

Ethernet connected.

Kent, Nick, Maxx(K,N,M) 4node-RPi-cluster:

CPU: Pi3B -A 1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM

4-node pi 3B cluster, 16 cores with 4GB of RAM

Ethernet connected.

Software

(a) Programming Language: Which one did you use? Why?

C language. Message Passing Interface (MPI) is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computing architectures.

I use the MPI library to write a benchmark for testing on distributed system, shared memory systems.

Reason: MPI is relatively easy to implement and debug.

(b) Also list the versions for the operating system and other software. This is useful if anyone tries to reproduce this work down the road.

VW-Haswell machine:(Linux version) $ lsb_release -a

Distributor ID: Debian, Description: Debian GNU/Linux 9.0 (stretch)

VW-Haswell machine: (MPI version) $ mpiexec --version

HYDRA build details: Version: 3.2

4. Project Overview and Results

(a) Write about what you did.

The experiment is including: Running the benchmark on several machines. Machines including shared memory systems and distributed systems.
Theory: Simpson's Rule, A better numerical integration algorithm than the rectangle rule, Converges more quickly.

$$f(x) = \int_0^1 \frac{4}{1+x^2}dx, a = 0, b = 1, n = 50$$

$$\int_a^b f(x)dx \approx \frac{1}{3n}\left[f(x_0) - f(x_n) + \sum_{i=1}^{n/2}(4f(x_{2i-1}) + 2f(x_{2i}))\right]$$

I apply this theory in writing it into serial code and change the sample frequency big enough so that it is suitable for parallel code. Then I set in the main function separate the integral range into number of nodes and let each node to integrate of each period and add up all the results.

(b) Write about any results you obtained.

I have obtained that with the serial code takes way longer time than the parallel code running on shared memory system and distributed systens. Best speedup is 13 on VW-Haswell machine, 97 on VW-Pi cluster, 9.19 on K,N,M-Pi cluster.

(c) If your project involves performance, give time and power comparisons if possible.

The benchmark on VW-Haswell machine result in finishing the job in 0.64s, speedup is 13.07, parallel efficiency is 0.4, number of digit accurate after decimal point. The parallel efficiency gets less than 0.9 when it is set to 4 nodes solving the problem.

On VW 24-node Pi cluster, its best hit is finishes the job in 1.04s, speedup is 96.97, parallel efficiency is 1.01. The parallel efficiency stays at 1 throughout when the node increases.

On K,N,M 4-node Pi cluster, its best hit is finishes the job in 4.86s, speedup is 9.19, parallel efficiency is 0.57. Its parallel efficiency drops less than 0.9 when it reaches 4 nodes.

(d) Pictures and graphs are useful, if applicable.

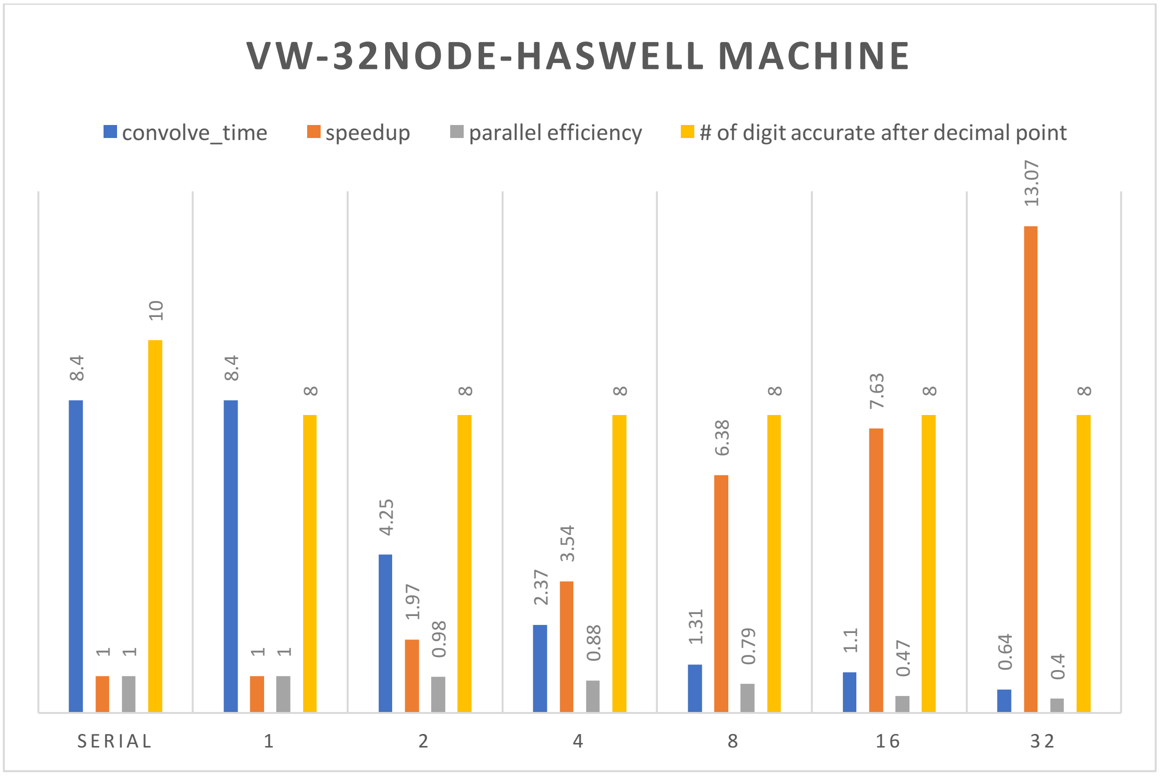

Figure 1. VW 32-node Haswell Machine, benchmark performance

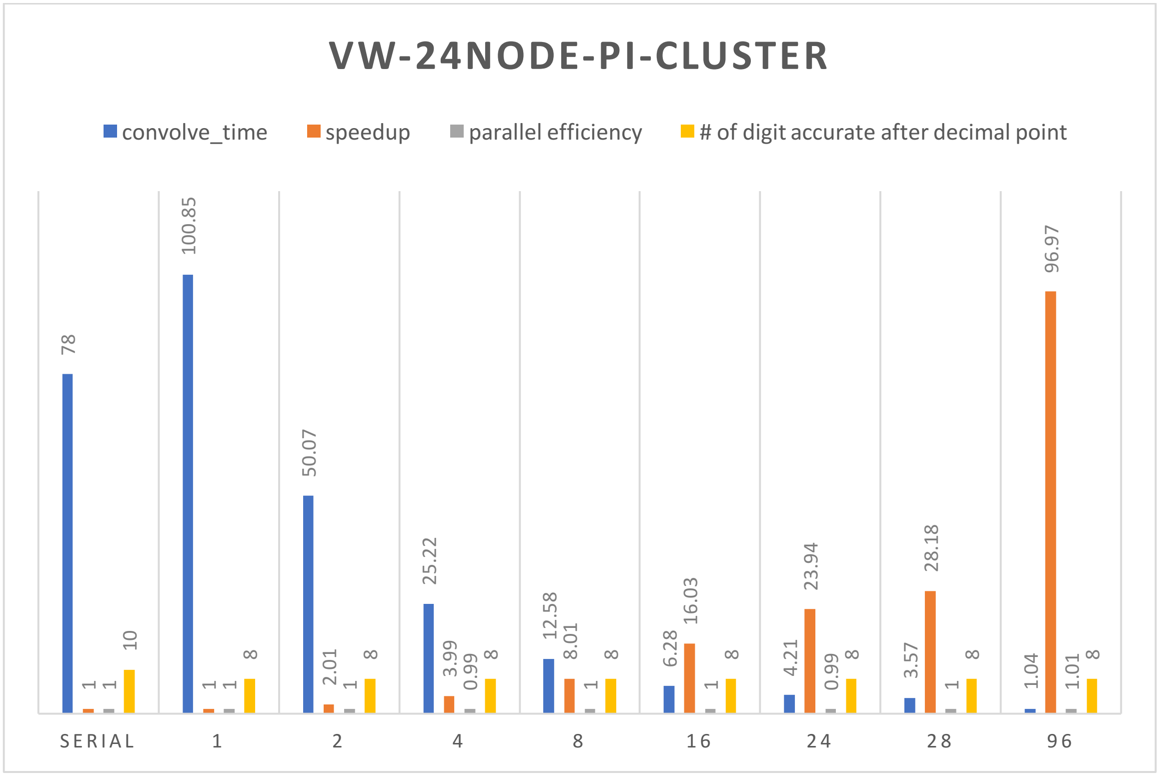

Figure 2. VW 24-node PI cluster, benchmark performance

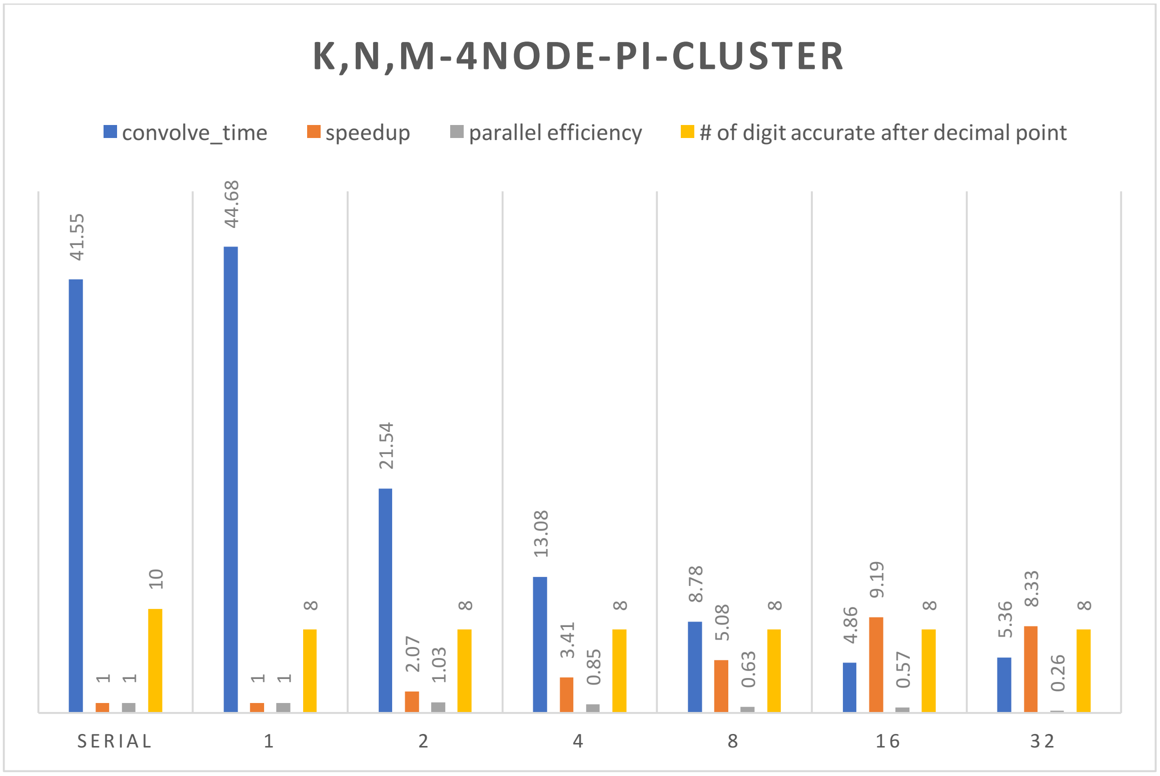

Figure 3. Kent, Nick, Maxx(K,N,M) 4node-RPi-cluster:, benchmark performance

(e) Feel free to write about things that *didn't* work. That's still of interest too.

I tried to add a remote node to the Pi-cluster, but the cluster is behind the firewall, kind of hard to reach this goal. Another problem is that the serial code is having higher precision 10 digits than the parallel code does 8 digits. This problem also shows when the parallel code when it runs on 1 thread. This parallel code bug would be noted and find later. I tried to measure the power of each node or each core when running the benchmark, but the power measure package needs a lot of data mining, with VW's domestic power measure package, it is more suitable to measure the total power of whole system while fully loading all the nodes. In general, power measure did not work.

5. Conclusion

Summarize what you did.

I write a benchmark in C language using the MPI library and run the benchmark on shared memory system, distributed systems and record the performance of these machines. The performance of the VW 24-node pi cluster shows a great scaling factor that the parallel efficiency stays at 1 with 96 threads running.

There is a bug in my parallel benchmark that is making the number of digit accurate after the decimal point is 8 and it's less than the serial code that is having 10 digits accurate after decimal point. I think this is some issue when the MPI that handles the double precision floating point data, it rounds off the last 2 digits of the data and cause this loss of 2 digit in accurarcy.

Future Work: List any improvements you might make if you had more time and resources to work on the project.

Step in some Differential Equation solving break into parallel coding. Write Fast Fourier Transform algorithm using parallel programing. Use GPU to parallel code.

Improve the serial code and the parallel code to get higher precision after the decimal point. One way is to change from using the Double Float data type to long array. The other way is to improve the formula that would promise a higher approximation to the PI.

Energy measure. In our shared memory system or the distributed system, it is hard to measure each core power consumption, but it is possible that we measure the whole CPU, I am planning on write a better benchmark and measure the CPU power consumption while letting the all the cores occupied.

6. Appendix

I allow project post on website.

7. Reference

[1] W.M, W.L 1967, Parallel methods for the numerical integration of ordinary differential equations, Math. Comp. **21** (1967), 303-320

[2] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George, 2007 Sep 21, Parallel global optimization with the particle swarm algorithm, Int J Numer Methods Eng. 2004 Dec 7; 61(13): 2296–2315.

[3] J. Sorenson, I. Parberry, 25 May 2002, 2 Fast Parallel Prime Number Sieves, Information and Computation, Volume 114, Issue 1, October 1994, Pages 115-130

[4] Bailey, David H., Borwein, Jonathan M., 04-22-2005, https://escholarship.org/uc/item/4281090t