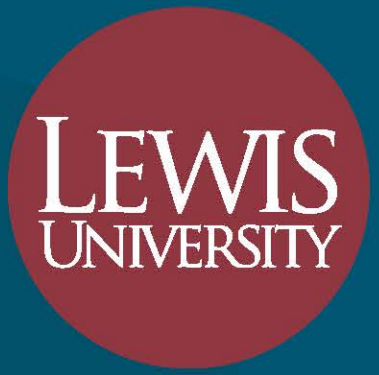
3/22/18



Implementing a High-Performance Computing Center

Quinn Stratton
Mentor: Dr. Ray Klump

Acknowledgements

- Prof. Clairardin
- Romeoville High School
- Dr. Livingston
- Dr. Schlachter
- Dr. Klump
- Dr. Powers
- S.U.R.E.

http://www.criticalcommons.org/Members/ccManager/clips/a-recalcitrant-supercomputer-defies-its-operator-in-willy-wonka-and-the-chocolate-factory/thumbnailImage

LEWIS UNIVERSITY
A Catholic and Lasallian University

Background

- **CPU (Central Processing Unit)** - "Brain" of computer, handles direct computations
  - handles one task at a time
- **GPU (Graphics Processing Unit)** – Also handles direct computations, but contains many smaller processors
  - inherently parallel
  - More computing throughput, more FLOP/s per Watt
- **Parallel Computing** – Computing that involves more than one processor working on the same problem in parallel

Speedup

- Linear speedup would be ideal
- Some algorithms are difficult to parallelize
- Any computer system will involve latency
  - Generally due to slow communications

Methods

- Cluster computing involves writing programs that run on multiple computers (**nodes**) in parallel
- *MPI* (Message Passing Interface): Used to implement *message-passing parallel programming model*
  - Allows for communication between nodes
- *CUDA*: Programming platform for developing on GPUs
  - Provides libraries for *C, C++, Fortran, Python, etc.*
- Not limited to these tools in general

Computation and Science

- Assist in numerical computation
  - Numerical solutions to differential equations, etc.
- Monte Carlo Simulation
  - Model random processes
- Machine Learning
  - Statistical inference from large data sets
- Computational Biology
  - *Human Genome Project*

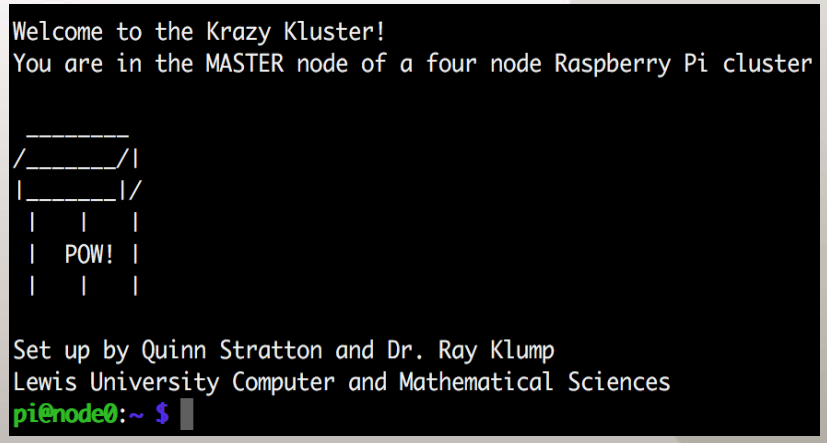
Goals/Motivation

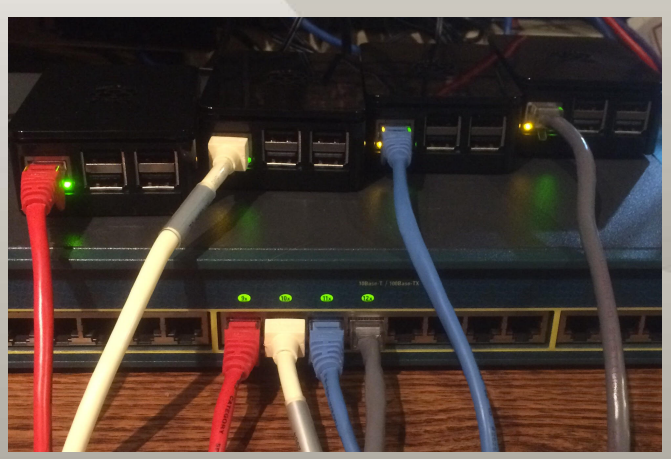
- Implement small-scale clusters from a low-level
- Computation is playing a greater and greater role in science
- Bring this power to Lewis

Raspberry Pi 3 Cluster

- Cluster of four Raspberry Pi 3 computers
- CPU-based
- Benchmarked at 2.222 GFLOP/s using HPL (High-Performance Linpack), 15.871 $\frac{MFLOP/s}{Dollar}$

https://www.raspberrypi.org/app/uploads/2015/08/raspberry-pi-logo.png

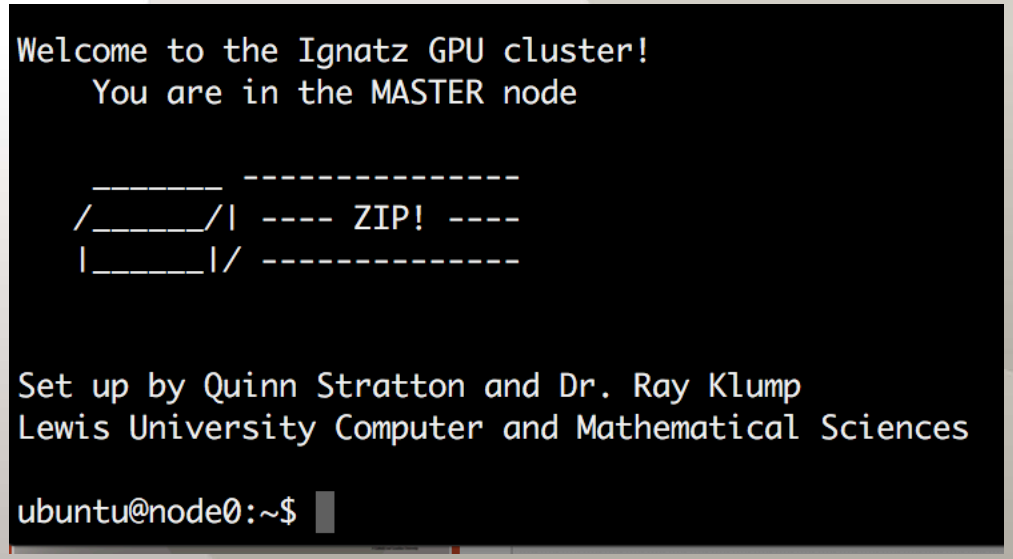




LEWIS UNIVERSITY
A Catholic and Lasallian University

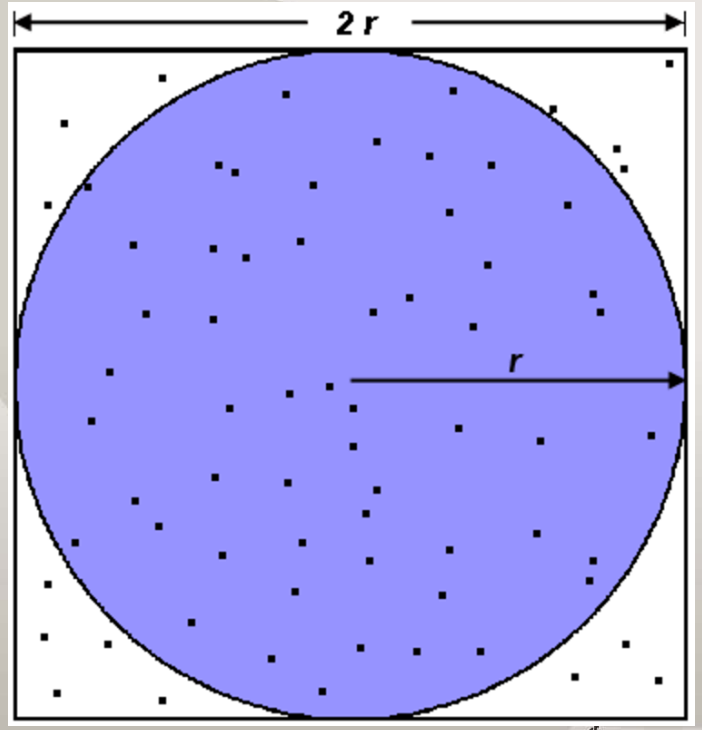
Jetson TK1 Cluster

- Cluster of four Jetson TK1 computers
- Each node has a CPU **and** a GPU with 192 cores



Case Study: pi_approx

- Monte Carlo method for approximating π
- Randomly generate N points within square
- Ratio of circle to square area is $\frac{\pi * r^2}{4 * r^2} = \frac{\pi}{4}$
- Number of points in circle (C) should be approximately $N * \frac{\pi}{4}$
- So $\pi \approx \frac{4 * C}{N}$



MPI only

```
// Initialize environment
MPI_Init(NULL, NULL);
// Find out rank and size
int world_rank;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
```

```
local_pi = 4.0 * (long double)local_count / (long double)local_N;

rc = MPI_Reduce(&local_pi, &pi_sum, 1, MPI_DOUBLE,
        MPI_SUM, MASTER, MPI_COMM_WORLD);
```

```
// Done with MPI
MPI_Finalize();

// MASTER process calculates value of pi using the global_count
if (world_rank == MASTER) {
        // Calculate pi value
        pi_approx = pi_sum / (float)world_size;
```

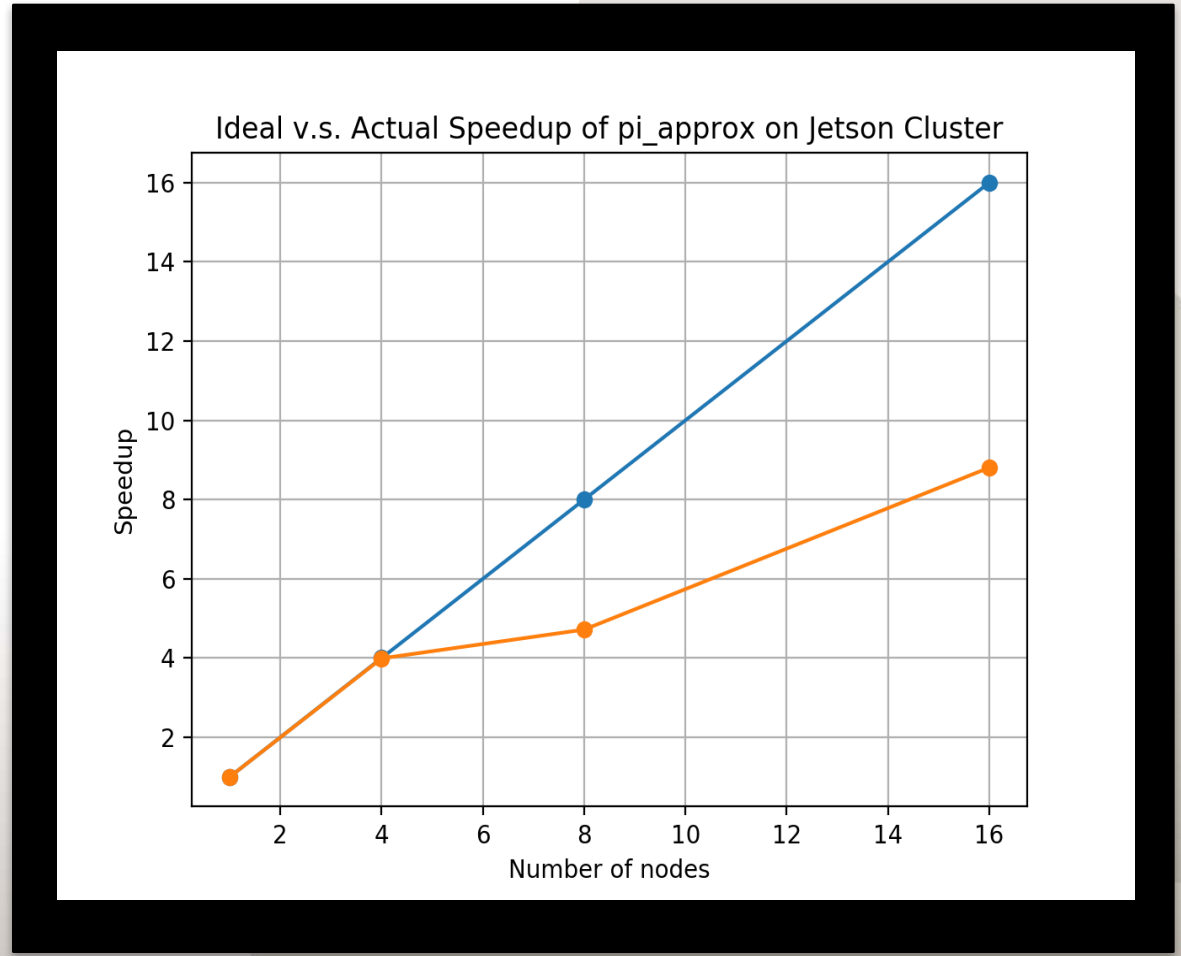
MPI with CUDA

- MPI handles splitting task up among nodes
- Each node now uses a GPU instead of CPU for processing

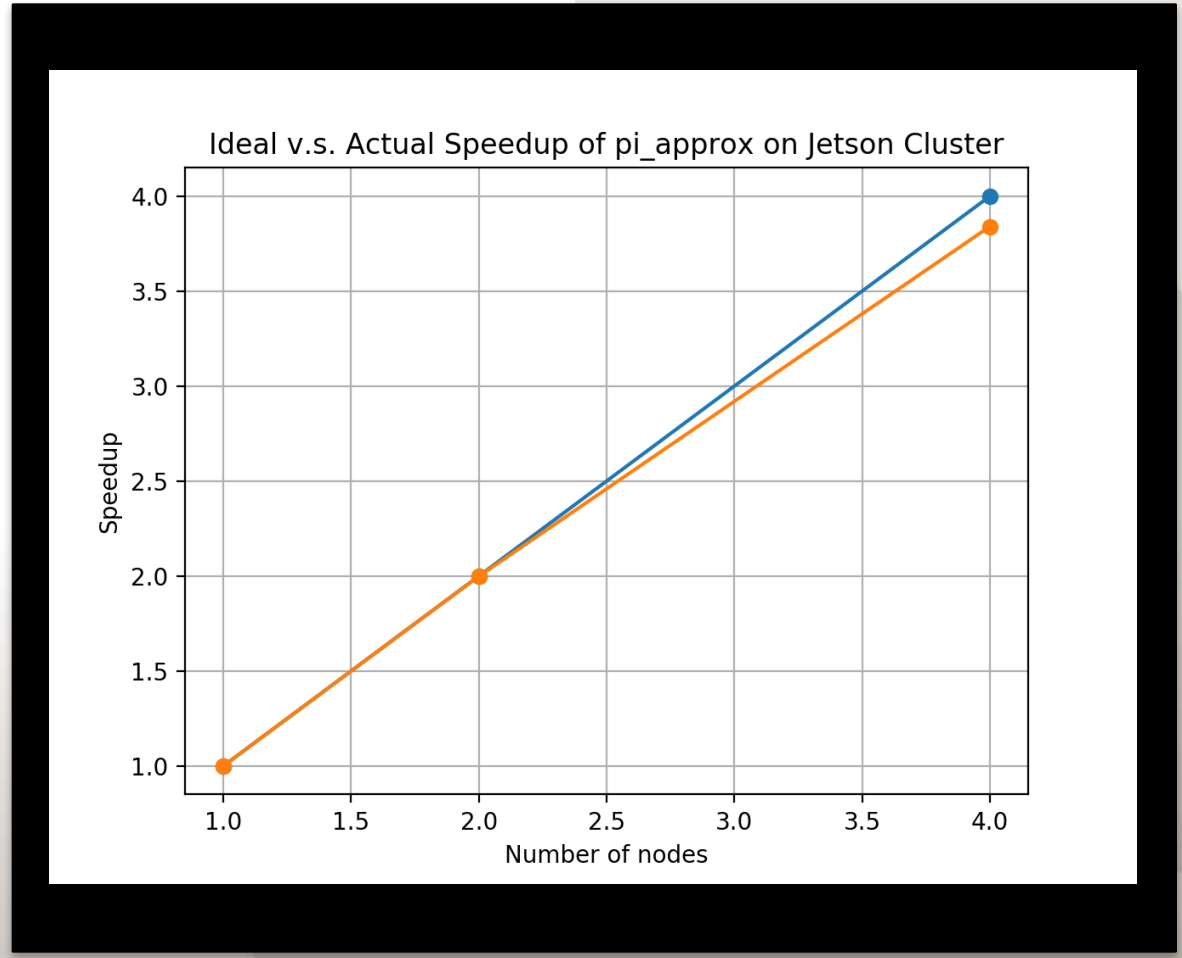
```
// Allocate memory on the CPU
int *in_circle = (int*)malloc(NUMBLOCKS*TPB * sizeof(int));
int *dev_circle;
// Allocate memory on the device (GPU)
cudaMalloc(&dev_circle, NUMBLOCKS*TPB * sizeof(int));
pi_kernel<<<NUMBLOCKS, TPB>>>(states, dev_circle);
cudaMemcpy(in_circle, dev_circle, NUMBLOCKS*TPB * sizeof(int),
        cudaMemcpyDeviceToHost);
```

Speedup of HPL on Raspberry Pi cluster

Speedup of pi_approx on Raspberry Pi Cluster

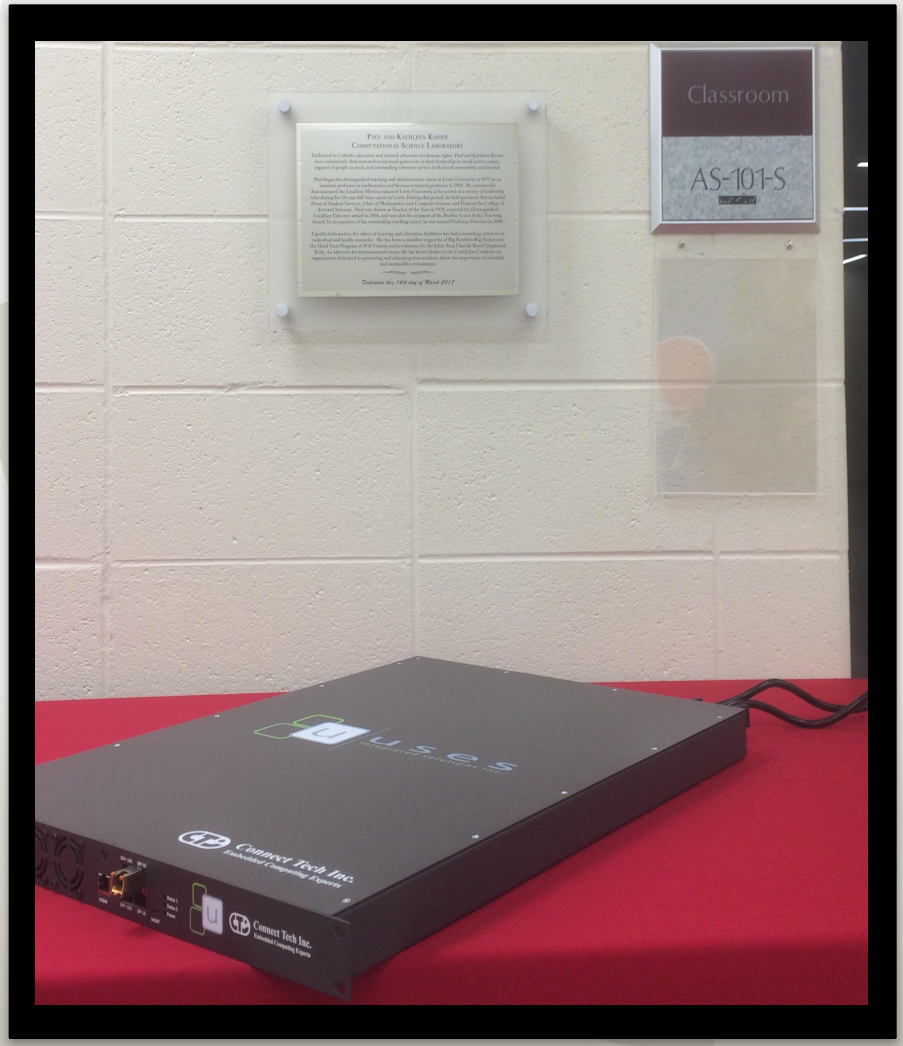


Speedup of pi_approx on Jetson TK1 Cluster



Further Inquiry

- Compare data to official Lewis University supercomputer
- TX2 Array Server
  - 24x NVIDIA Jetson TX2
  - GPU: 1 TFLOP/s 256-core with NVIDIA Pascal Architecture
  - CPU: 4x 64-bit ARM A57 CPUs + 2 NVIDIA Denver CPUs
  - Memory: 8GB LPDDR4
  - Storage: 32 GB eMMC



Even Further Inquiry

- Explore different programming models for high-performance computing
  - Python
  - Matlab
  - Functional languages
- Collaboration outside of Computer and Mathematical Sciences

Takeaways

- Intersection of computation and science is a major aspect of current research across disciplines
- Important to be familiar with these methods
- Allows for very interesting opportunities for interdisciplinary study and collaboration

Appendix

- Code, data, and documents
  - https://github.com/drVulter/Supercomputer2017

References

- http://www.netlib.org/benchmark/hpl/
- https://www.top500.org/
- https://portal.tacc.utexas.edu/c/document_library/get_file?uuid=e05d457a-0fbf-424b-87ce-c96fc0077099&groupId=13601
- http://www.instructables.com/id/How-to-Make-a-Raspberry-Pi-SuperComputer/
- https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/
- https://computing.llnl.gov/tutorials/mpi/