A Recipe for Estimating π

Imagine that we have a dartboard, which is made of a circle that just fits perfectly with in a square. If you throw enough darts at the dartboard, you will cover it completely. The number of darts you can fit depends on the size of the dartboard. The fraction of total darts that land in the circle for the dartboard compared to the number that fit in the whole square will be the same as the fraction of the area that the circle takes up in the square. The area of the circle is (π radius^2) and the area of the square is (2 radius)^2. From these we can get a recipe for estimating the value of π that only requires us to know how many darts landed in the circle and how many darts were thrown in total.

Concepts needed:
Math

1. π : the number of times you can wrap a circle's radius around half the circle. π allows you to calculate the area and the circumference of the circle if you know the length of the radius.
2. Area of circle
3. The area of the circle (π radius^2), Area of a square : lenth^2
4. Proportion
5. Ratio

Parallel Computing
Many hands make small work if . . .

1. The work is well organized
2. The time spent organizing the work and doing it by many workers is less than the time spent by one worker to do it all.

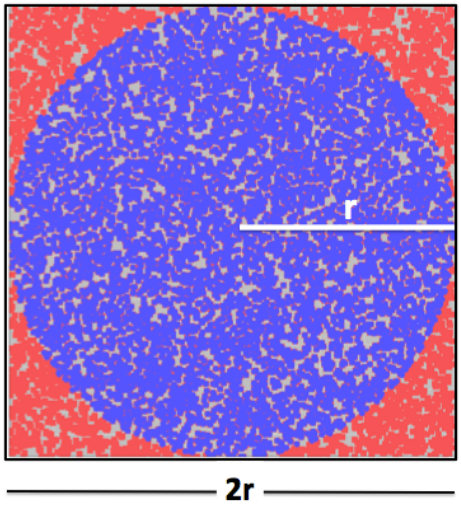

Figure 1: A "dartboard" made of a circle inscribed in a square.

For this dartboard, the circle fits perfectly just inside a square—the circle is inscribed in the square.
The area of the circle is πr^2 , where r is the length of the circle's radius.
The area of the square is 2r *2r

We want to see if we can calculate the value of π by playing darts!

A ratio is a comparison. For example, if we want to know how much of the square's area the circle takes up, we can divide the area of the circle by the area of the square.

Ratio of areas = $\frac{\pi r^2}{2r * 2r} = \frac{\pi}{4}$

If you were to throw enough darts at the board, they would cover all the area in the square and circle. The number of darts the circle would depend on how big the circle was. Another way to say this is: The number of darts the circle can hold is proportional to the area of the circle.

The same is true for the whole square (counting all the darts in the circle too since all of the circle is inside the square). The number of darts the square can hold is proportional to the area of the square.

The ratio of the number of darts in the circle to the total number of darts in the square will be proportional the ratio of the area of the circle to the area of the square.

$$\frac{\#darts\ ciricle}{\#darts\ total} \cong \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

So that means that

$$\pi = 4\ \frac{\#darts\ ciricle}{\#darts\ total}$$

We are going to play 20 games of darts where we throw 20 darts in each game. So 400 darts will be thrown total. We will assume that this is like throwing 400 darts at the same board. Then we will count the darts that land in circle and calculate π.

$$\pi = 4\ \frac{\#darts\ ciricle}{\#darts\ square} = 4\ \frac{\#darts\ ciricle}{400} = \frac{\#darts\ ciricle}{100}$$

The Computer

A supercomputer gets its processing speed by dividing up the work of calculations between many processing elements.

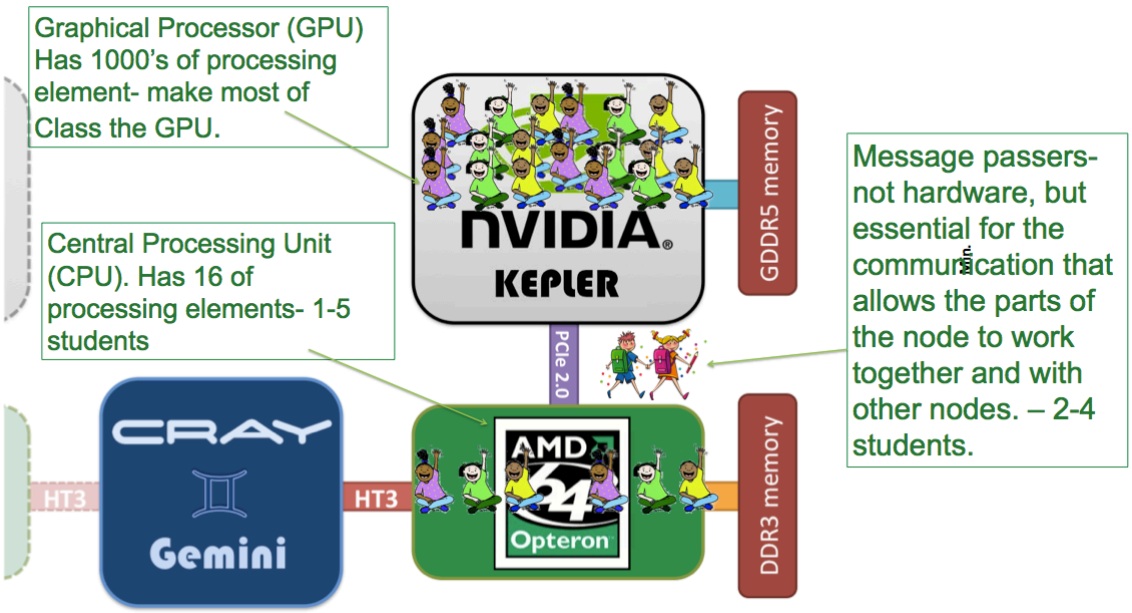

Figure 2. One Titan Node. [1][2]

For a Titan node this means that the work is spread over the central processing unit (CPU) and graphical processing unit (GPU). GPUS were originally designed to shade pixels on computer monitors. Which means that the GPU is a processor that is specialized to do hundreds of simple calculations, like adding two numbers, all at once. Each GPU has 192 "workers" that can each do a simple calculation. The CPU can also so simple additions very fast but it only has 16 workers. Before there were computers like Titan, the CPU had to do all the work, However on Titan, the CPU uses the programmer's instructions to farm work that can be done at the all at the same time (in parallel) to the GPU. The work that is sent called a kernel. Titan has 18,688 nodes

We are going to race the CPU and GPU to calculate π.

The Program

A kernel is a set of instructions, generally part of the program intended to execute on a specific set of processing elements.

10-20 Students will be the GPU. Each GPU will get a sheet with a dartboard that has 20 darts. Count the number of darts in the circle.

One or two students will be messenger(s) who collect all the work. Messengers must carry an array along (a sheet with 20 line) so each processing element of the GPU can write down the number of darts that were counted.

Reduction: The 20 counts from the boards will need to be summed together and then divided by 100. This is called a reduction. The students should spend a few minutes discussing how their processor should do this. If only one student does it, the process is "serialized". If multiple students do the sum there is a chance take better advantage of the parallelism of the processor, but the student will need a plan for communication and error checking.

1 to 4 students will be the CPU.

The CPU will do the same counting but it will only have fewer workers to count all the darts in the circles on the 20 sheets. It will also have less organization to do to collect and sum the work for the reduction.

Who should win?

If the GPU can get its work collected and reduced efficiently, it should win- since each GPU processing element has a small number of sheets to count. If they cannot organize the reduction, or they can get the parallel work (counting the darts in the circle) done fast enough to make up for the time it takes to distribute and collect it, the CPU may win due to the fact that it has less challenge organizing the work.

Coding Exercise

This is one the example problems that the Oak Ridge Leadership Computing Facility (OLCF) uses to teach its uses how to use Titan. If you would like extend this to teach C , MPI or OpenMP, you can use the OLCF tutorial,
Serial to Parallel: Monte Carlo Operation,
https://www.olcf.ornl.gov/tutorials/monte-carlo-pi

For Human Laptops

Don't have a need to illustrate a supercomputer with GPUs?

You can configure the students into any computational hardware. For example, this can be refactored to show exactly why hexacore laptops perform faster than dual-core laptops. While this may seem obvious, the students will still have to talk about how to organize the tasks with which that are about to be programed. Group the students into sets of 2 CPU laptops versus 6 CPU laptops.

Programs for Younger Human Supercomputers

For younger students, programing them to make 40 origami rabbits [4] is a good choice because rabbits can be made in parallel, but the paper distribution and collection of finished rabbits are waiting points that students can discuss how to mitigate.

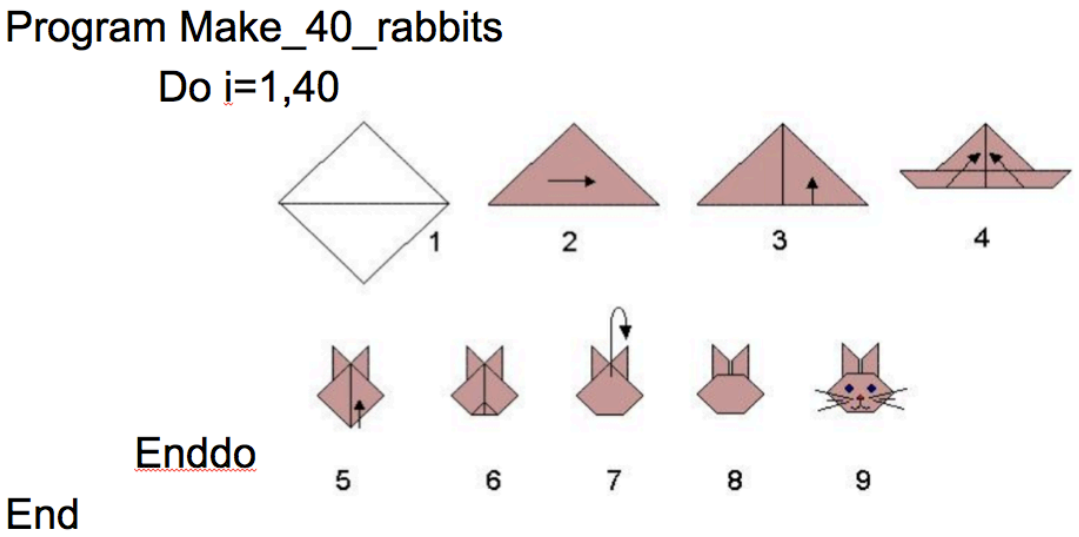

Figure 3: A program is just a set of instructions, like those used to make origami. [4]

Materials: 80 sheets of Origami paper or other paper that has been cut into a square shape.

Again for this exercise organize students in to a CPU, and GPU and messengers, with the largest number of students as the GPU. Messengers must pass out the paper and collect the Rabbits.

To add complexity, make some of the students in each processor be error checkers. Give each of them a well-made rabbit and have them compare it to the ones made by the students in the processors. The students can choose how many error checkers there will be in each processor, but those processing elements can not make rabbits that contribute to the total of 40.

References

1. The Oak Ridge Leadership Computing Facility www.olcf.ornl.gov/support/system-user-guides/accelerated-computing-guide
2. Clip Art: http://www.clker.com/clipart-students.html

3. https://www.olcf.ornl.gov/tutorials/monte-carlo-pi

4. The Origami resource center. http://www.origami-resource-center.com