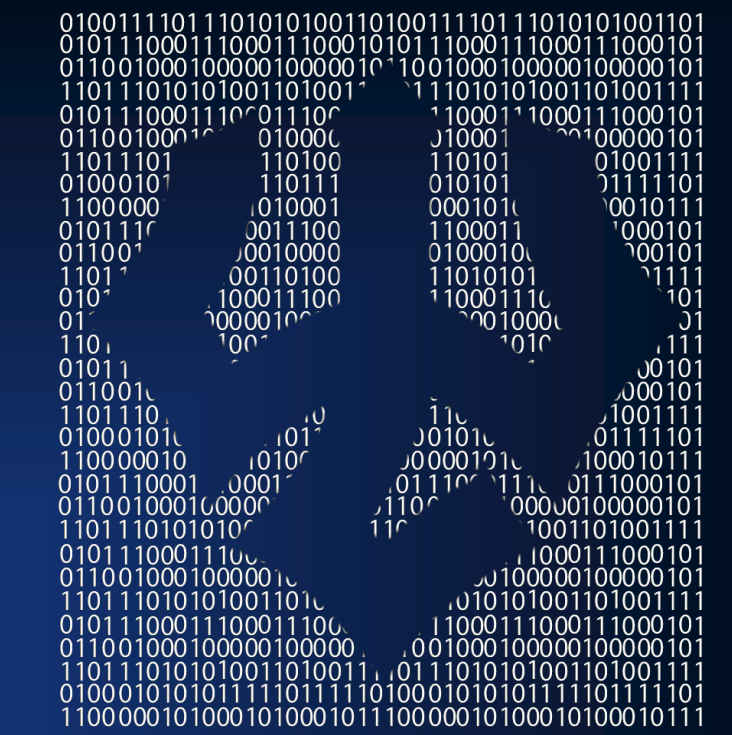




Python Parallelism in CS2/DS

Dr. Joshua Stough
Washington and Lee University



Module Goal: Readdress Divide & Conquer with Parallelism

CS2/DS @ W&L

- Goal: efficient interaction with and manipulation of collections of stuff
- PDC applications: sorting, recursive tree structures, image processing,...
- Python **multiprocessing** module:
 - **Process/Pipe**: spawn Processes for smaller problems, communicate through Pipe
 - **Pool/map**: pool of worker processes if no communication required

Module Code Available Online

- All code includes timing versus sequential
- Using **Pool/map**:
 - Shown: parallel integration
 - Developed in-class: sum of primes
 - Assigned: Monte Carlo simulation
- Using **Process/Pipe**:
 - Shown/Developed: parallel quicksort
 - Assigned: parallel mergesort



Evaluation

- Positives – Python is easy!
 - Little syntactic complexity
 - Students quickly see impressive speedup
- Still needed:
 - Better integration into course
 - Client/Server app
 - Synchronization

Pool/map Example: Monte Carlo pi

- Generate N points in the unit square
- Fraction of points also in unit disk approaches $\pi/4$
- Parallel Solution:
 - Instantiate **Pool** of P processes
 - Have each estimate pi on N/P samples
 - Return the average
- Speedup linear in number of cpus

```
import random, time
from multiprocessing import Pool, cpu_count

def montePi(num):
    """Return an estimate of pi based on num iter."""
    numInCircle = 0
    for i in range(num):
        if random.random()*2 + random.random()*2 <= 1:
            numInCircle += 1
    return numInCircle

N = 10000000
cpus = cpu_count()
args = [N // cpus] * cpus
#Distribute extra work if work cannot be evenly distributed.
args[:N%cpus] = [x + 1 for x in args[:N%cpus]]

start = time.time()
pool = Pool(processes = cpus) #instantiate the Pool
estimates = pool.map(montePi, args) #Compute cpus pi's
result = sum(estimates)/len(estimates)
elapsedTime = time.time() - start
print('pi estimated at %f in %f sec.' % (result,elapsedTime))
```

Process/Pipe Example: Mergesort

- Mergesort the left half-list
- Mergesort the right half-list
- Merge the two sorted half-lists
- Parallel Solution:
 - Mergesort the left and right half-lists *concurrently*: spawn new Processes
 - Merge the results of the subprocesses
 - Use **Pipe** for return values.

```
from multiprocessing import Process, Pipe
import time, random

def merge(left, right):
    """returns the merge of the two already-sorted lists."""
    #...

def mergesort(lyst):
    if len(lyst) <= 1:
        return lyst
    ind = len(lyst)//2
    return merge(mergesort(lyst[:ind]),mergesort(lyst[ind:]))

def mergeSortParallel(lyst, conn, procNum):
    """Mergesort the left and right sides in parallel, then
    merge the results."""

    if len(lyst) <= 1:
        conn.send(lyst)
        conn.close()
        return

    if procNum <= 0:
        conn.send(mergesort(lyst)) #Call non-parallel
        conn.close()
        return

    ind = len(lyst)//2

    pconnLeft, cconnLeft = Pipe()
    leftProc = Process(target=mergesortParallel, \
        args=(lyst[:ind], \
            cconnLeft, procNum - 1))

    pconnRight, cconnRight = Pipe()
    rightProc = Process(target=mergesortParallel, \
        args=(lyst[ind:], \
            cconnRight, procNum - 1))

    leftProc.start()
    rightProc.start()

    conn.send(merge(pconnLeft.recv(), pconnRight.recv()))
    conn.close()

    leftProc.join()
    rightProc.join()
```

Washington and Lee & Computer Science

- ~1800 students, undergraduate only
- CS Dept:
 - 4 faculty
 - 5-10 majors, 1-5 minors per year
 - Language, Web Apps, Linguistics, Image Analysis
- Thanks: NSF/TCPP Early Adopter award

