"Big Data Algorithms"
~ Talk Outline ~

To profit from Big Data requires a new approach. Many of the old algorithms and old techniques do not scale to the new world of Big Data. We need new ways of doing things.

This is a preparatory talk for the MFE "Big Data in Finance" (BDiF) course, and as such it introduces material that will be useful for those students hoping to take (and pass!) the course. In a two hour talk, the topic of "Big Data Algorithms" cannot be covered in-depth so I have chosen to focus on those algorithms and topics that are most useful to successfully completing the BDiF course.

1. The right answer? Or a "good enough" answer?
    a. Speed versus accuracy.
    b. Being 51% right is a mark of genius?
    c. Making intelligent bets.
2. Moore's Law and Algorithms
    a. Moore's Law and hardware gains in computational power.
    b. Algorithm development compared with Moore's Law over the same period.
    c. Amdahl's and Gustafson-Barsis' Laws.
    d. Parallel algorithms.
    e. Approximations and heuristics.
3. Big-O Notation
    a. The need for Big-O notation.
    b. A compendium of Big-O performance measures for common algorithms.
    c. Sublinear behavior: in space, in time, in communication.
    d. P and NP problems. Why does nature find NP problems easy?
4. Introducing the CUNY Supercomputer Cluster
    a. Student access to compute and data resources on the CUNY cluster.
    b. Compiling and running programs to run at scale on the cluster.
    c. Running distributed programs on the CUNY cluster.
    d. Running distributed multi-GPU programs on the CUNY cluster.
5. MapReduce and Hadoop
    a. MapReduce as a divide-and-conquer algorithm.
    b. MapReduce on Hadoop.
    c. MapReduce on CPUs and GPUs in a few lines of code.
6. Monte-Carlo on GPUs
    a. GPUs and SIMD.
    b. Full Monte-Carlo on GPUs with a single line of code!
7. Some other random thoughts and techniques
    a. Sketching and streaming. Extremely small-space data structures that can be updated on the fly in a fast-moving stream of input.
    b. Counting distinct elements. Min-hashing.
    c. Slime mold algorithm. Nature knows best!
8. What's it gonna be? More data + simple algorithms? (Pipelining?) Or less data + sophisticated algorithms?