- You do not need “Bigdata” most of the time, just think about where are your bottlenecks
- Assume just batch processing for now and that we do not want to change algorithm
- Where is your bottleneck?
  - CPU
  - Data movement?
  - Always measure.
- My Python program is too slow, and I want to run single threaded
  - Properly vectorize (numpy, no iter_rows, …)
  - Numba
  - Cython
  - C++/Rust Bindings
- Memory/disk bottlenecks
  - Compress data
    - Float64 -> float32
    - NPZ format with compression (DEFLATE)
      - Even better are snappy/zstd
  - Get bigger RAM (so your data will fit in it)
  - Or SSD instead of HDD
  - Compressed data in RAM, which you decompress on demand can be better than uncompressed on disk (YMMV)
- I want to use whole CPU
  - Multiprocessing
  - Or carefully running multiple programs
    - Remember MAD (grid engine)
- I want to use GPU
  - Pytorch for ML
  - Or Cupy
  - Or go low level and cry
- I want to use multiple machines
  - Carefully running multiple programs
  - Dataflow (remember MAD)
- More machines, more problems (if machine will fail in apx. 1000 days, then when you have 1000 machines some will fail that day). It is similar as with organizing people gatherings:
  - 1 person: easy to arrange.
  - >2 persons: coordination.
  - >10 persons: requires leader in charge.
  - >100 persons: requires fixed menu.
  - >1000 persons: no one knows many people.
  - >10,000 persons: too few hotels for most cities.
  - >100,000 persons: someone will die that day

- What about some processing with state (e.g. streaming or web app)
  - Having distributed state is PITA

    - What is your state?
    - What is stateless processing?
    - Maybe you can hold state in DB (or even Redis) and just have multiple stateless workers
        - Cloud providers handle this easily