Sketching Streams

Chris Taylor
DoD

Overview

- What-Why Sketch?
- Sketches
  - Hyper Log Log Sketch
  - Frequency “Heavy Hitter” Sketch
  - Quantile Sketch
  - Theta Sketch

What-Why Sketch?

What-Why Sketch?

- Data sets exceed traditional commodity compute capabilities
  - Static and Streaming data
  - Data set is “noisy” (biology, physics)
- Approximate results have value

What-Why Sketch?

- Compute dynamic “summaries” of a dataset according to a predefined set of computational constraints
  - Storage size
  - Accuracy, precision...user provided tolerances
- Sketches are “monoidal” in nature; satisfying a suite of set operations (union, difference, etc)
  - Functional programming concepts
  - Parallel prefix summarization

What-Why Sketch?

- “Data analytic” platforms adopting sketches
  - Yahoo's “Data Sketching” library
  - Druid integration with Yahoo's library**
  - Redis support
  - Several opensource projects for Spark/Hadoop

**** Traditional Database, “Columnar” Stores, “Big Table” Database**

What-Why Sketch?

- Measuring Performance
  - Using Chapel 1.15!
  - **Measured sketch update performance**
  - Each algorithm receives a randomly filled array of 100K integers
  - Each algorithm provided 5 minutes to 'add' or 'update' a sketch (serial loop) over sets of the 100K integers
- Results are the total number of 100K block-integer updates completed in ~5 minutes

HyperLogLog

HyperLogLog

- Philippe Flajolet
- Analyzes a stream of hashed values (bit-pattern observables)
  - Split each hashed value into *m* sets
  - Collects “runs” of zeros for each *m* set
- Provides a Stochastic Average using collected bit-pattern information
  - Compute a harmonic mean of each *m* bit set (for each new value)

HyperLogLog

- Hashed Value:

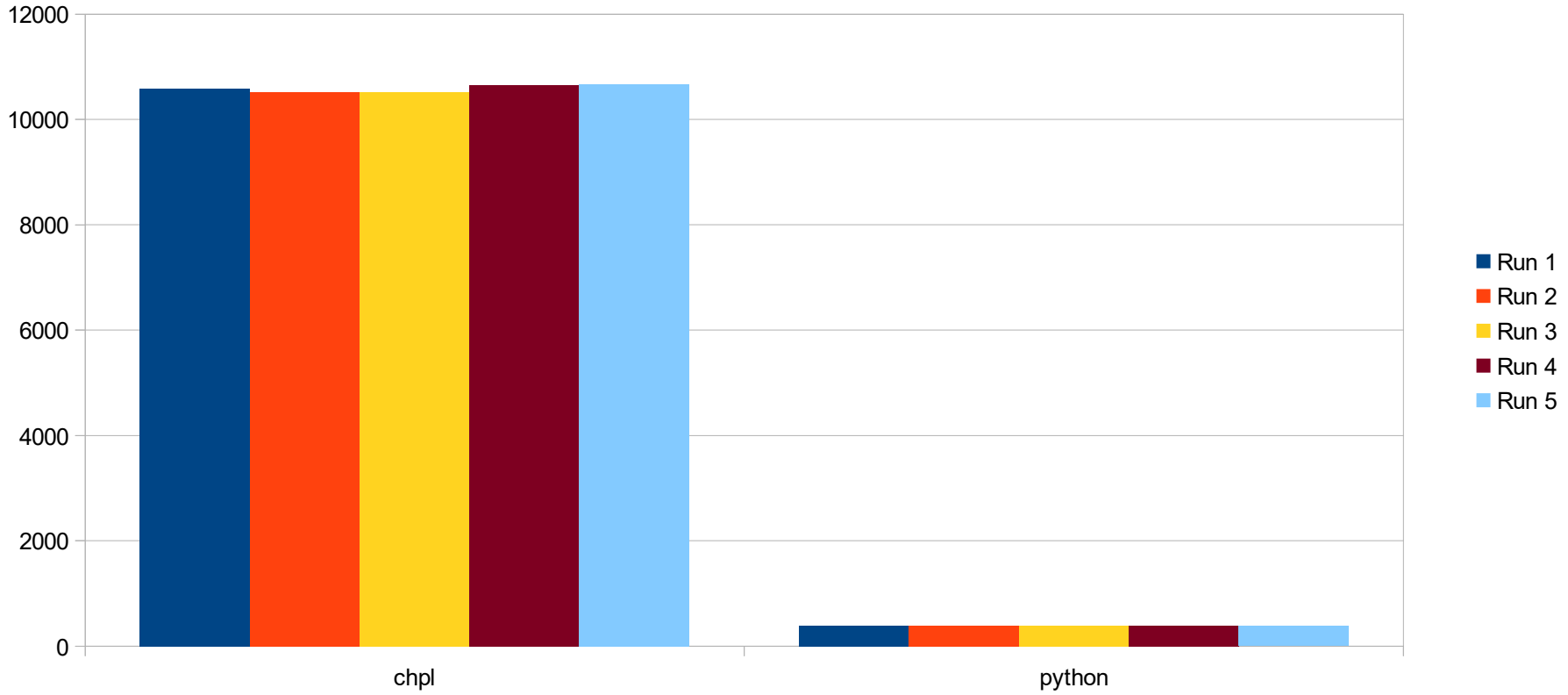
  000011000111

- Split hash into bit-pattern sets (*m*=3):

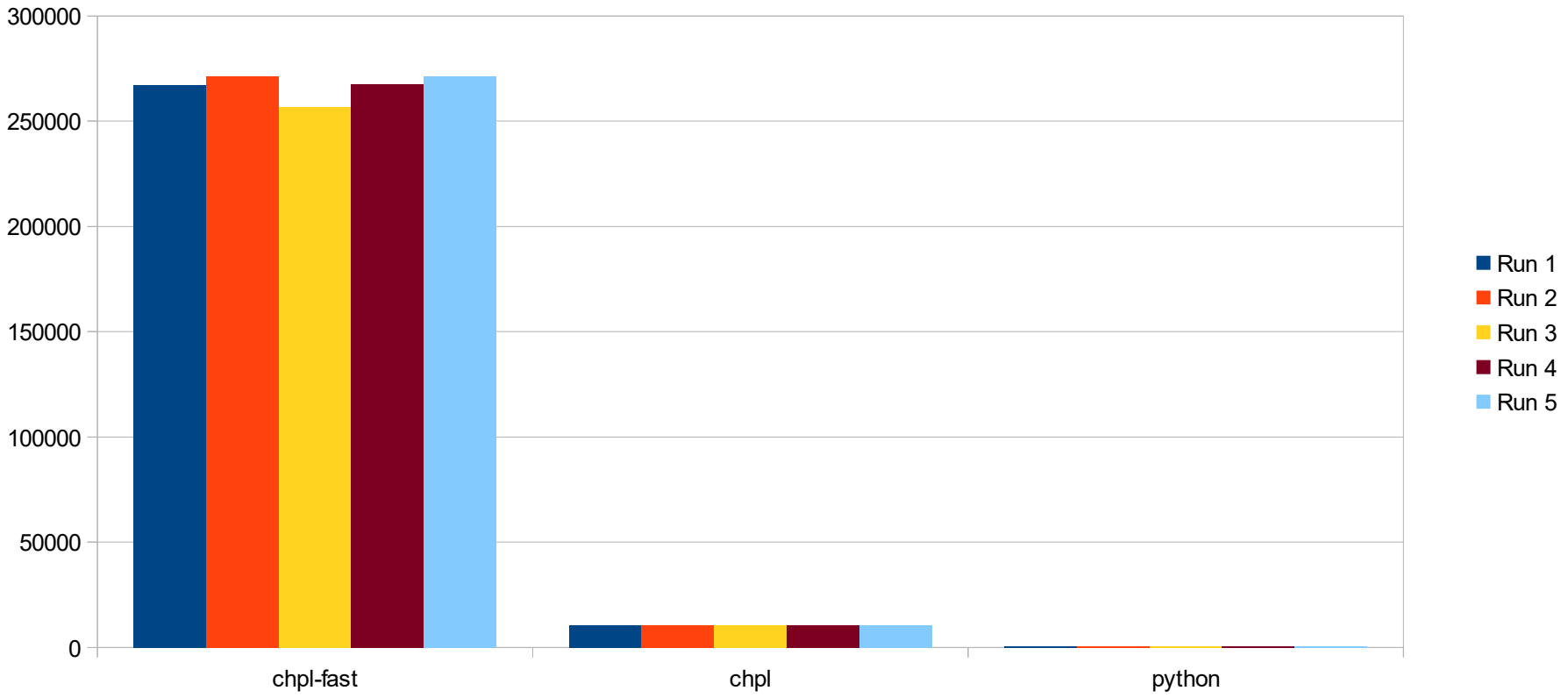
  [[000], [011], [000], [111]]

- Compute running harmonic average over existing bit-pattern sets

HyperLogLog



HyperLogLog

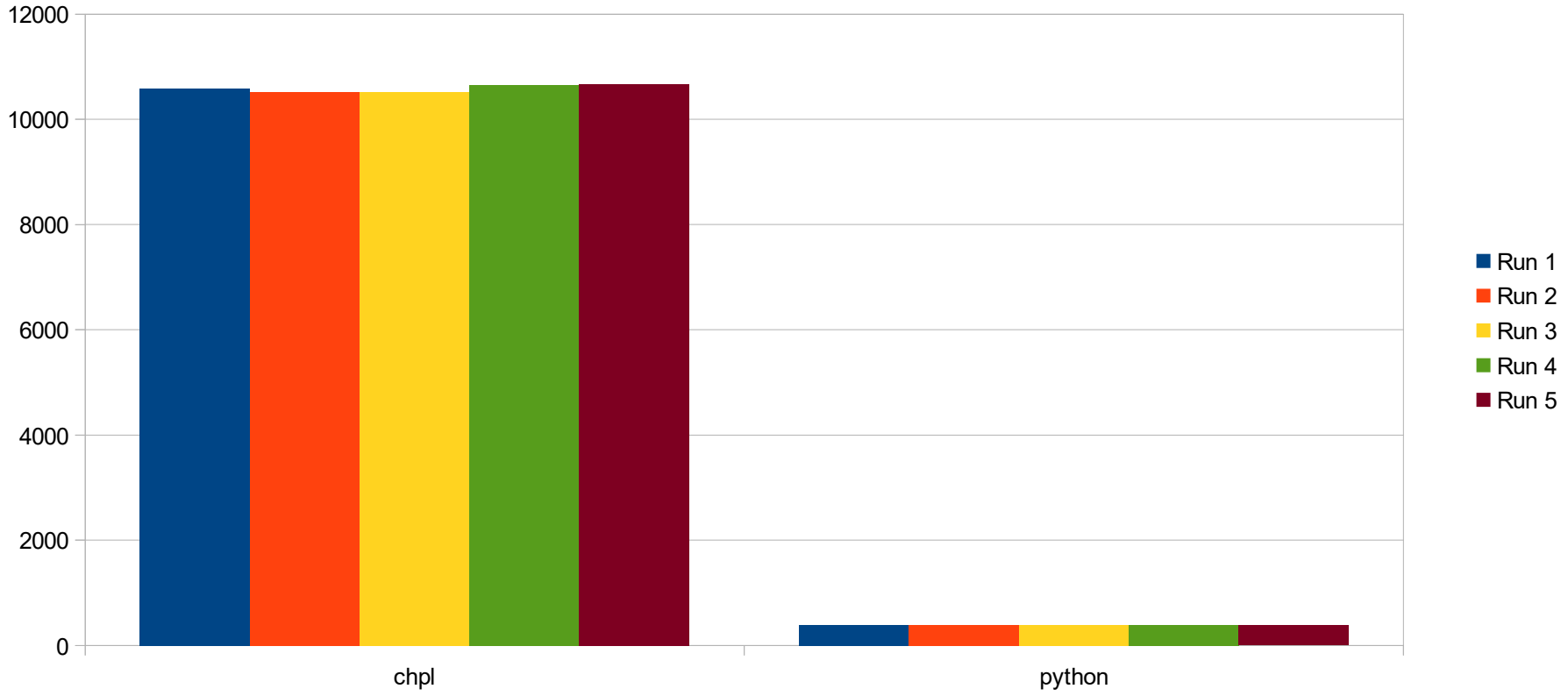


Frequency Sketch

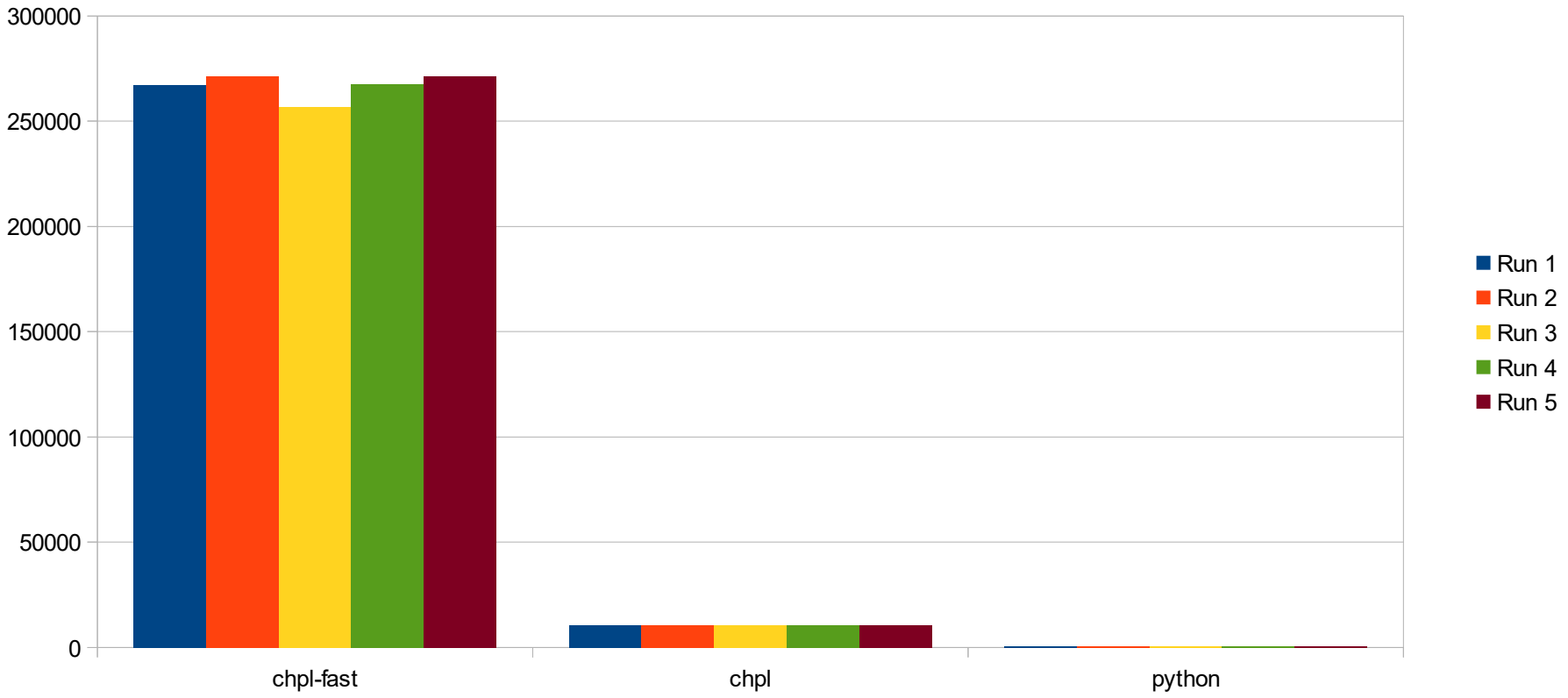
Frequency Sketch

- Implementation of Misra-Greis Algorithm
- Stores k-1 (item-counter) pairs as a set
- If a new item is in the set's range
  - Increment a counter
  - Else find an empty counter, add item, and set counter to one
- Decrement all k-counters if all counters have been allocated
- Over time, low frequency elements are removed, making space for higher frequency items.

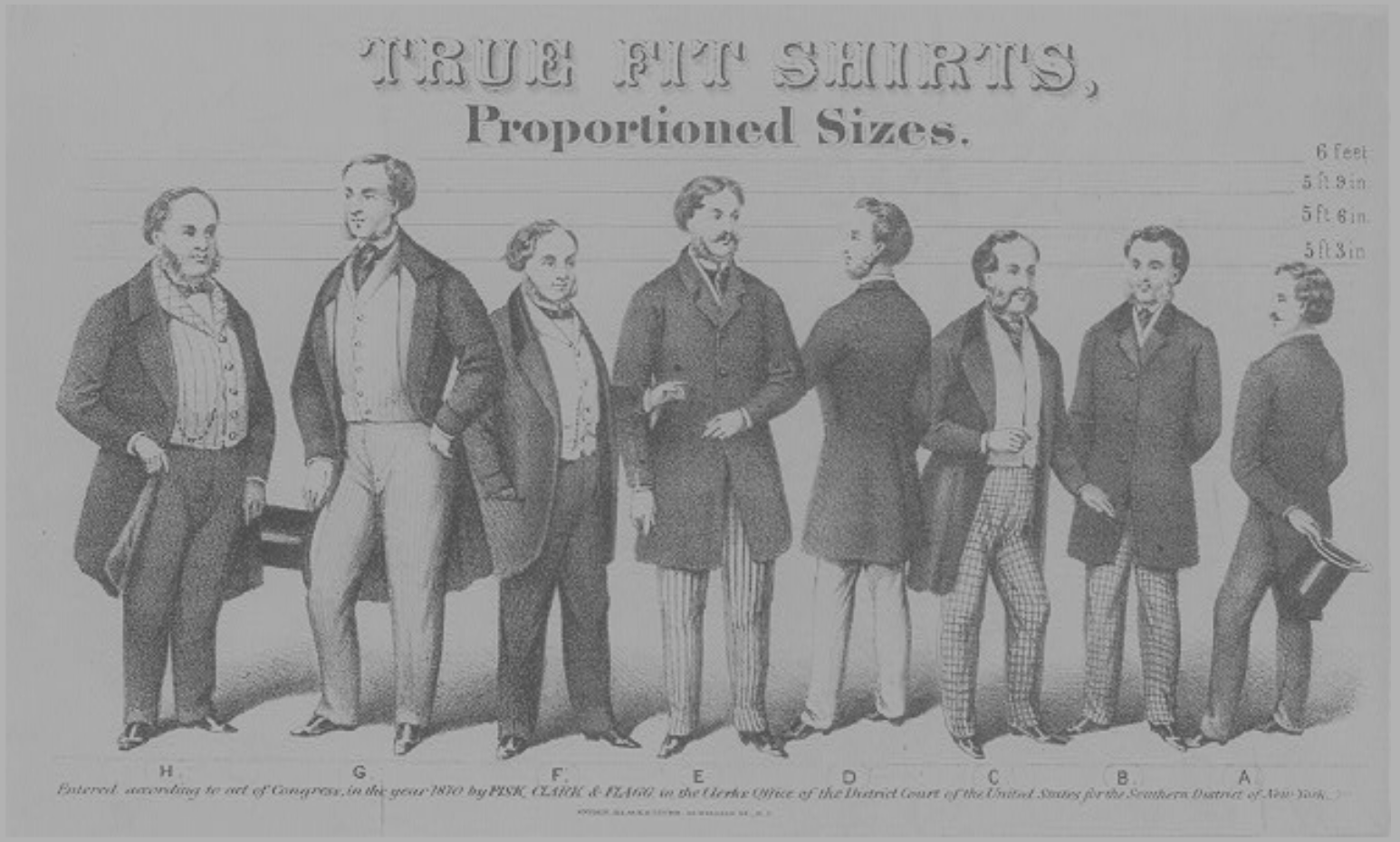
Frequency Sketch



Frequency Sketch



Quantile Sketch



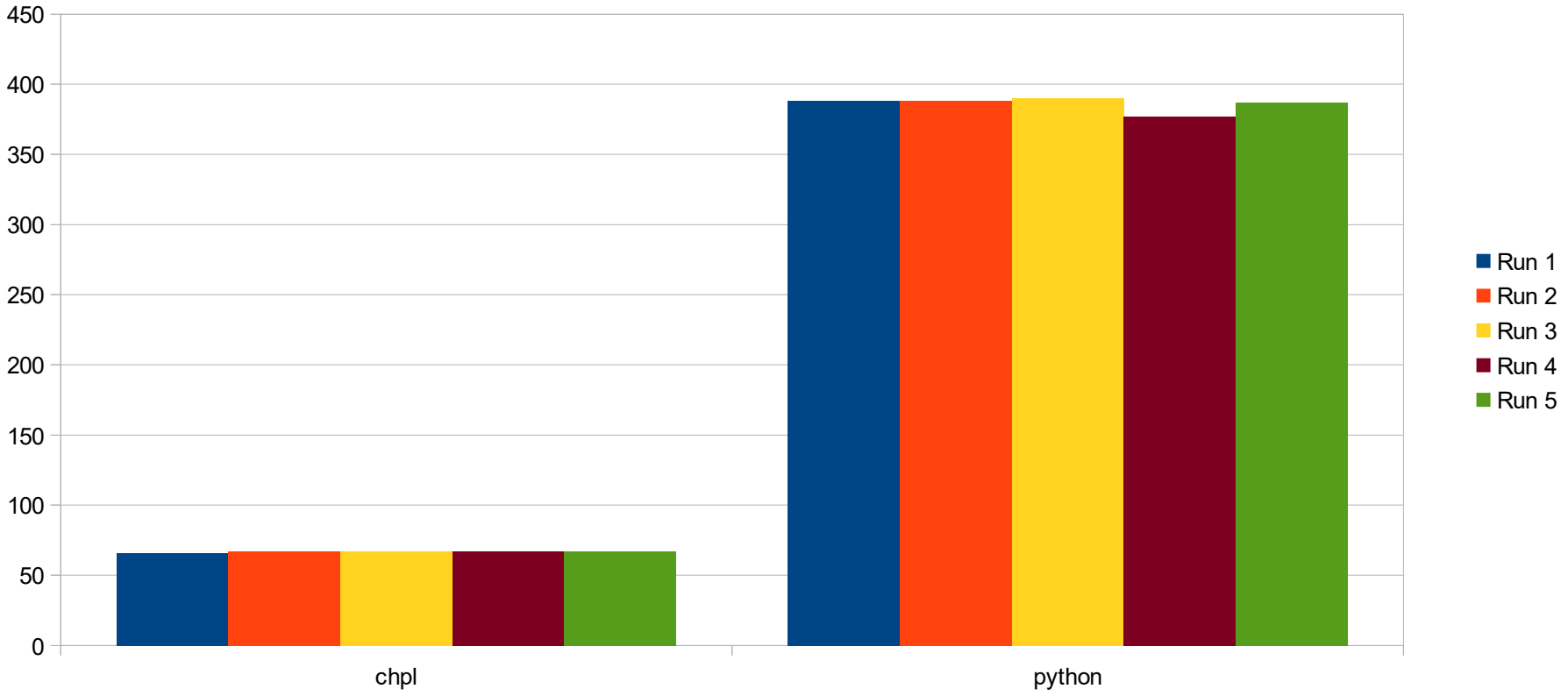
Quantile Sketch

- “Low Discrepancy Mergeable Quantiles Sketch” (Agarwal, Cormode, Huang, Philips, Wei, Yi)
- Non-deterministic!
- Select elements (upper/lower bounds) from the stream under a rank constraint:

  normalized rank: i|S|/k for 1 <= I <= k ~= 1/e

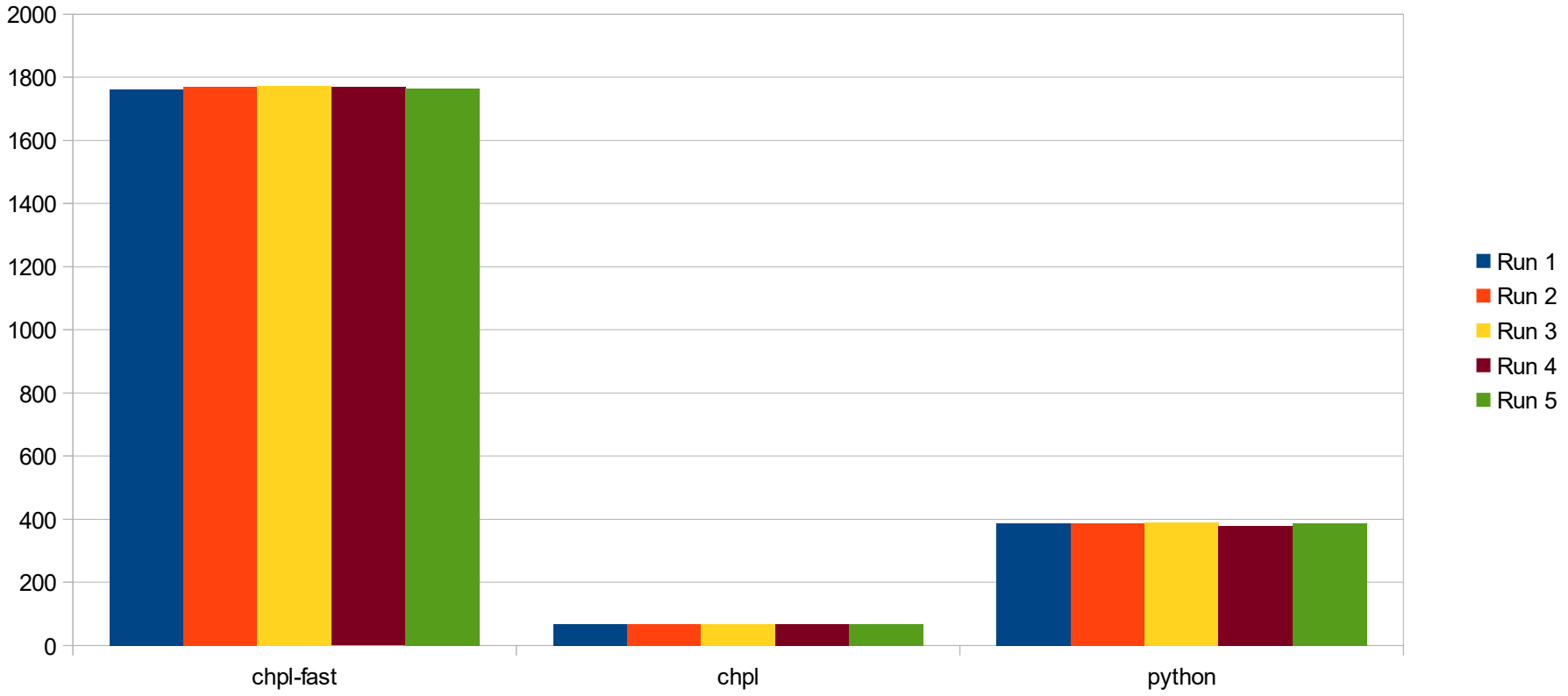
- Using the selected elements, or summary, compute quartile information.

Quantile Sketch



**** Chapel has to perform several domain resizes, could use optimization**

Quantile Sketch

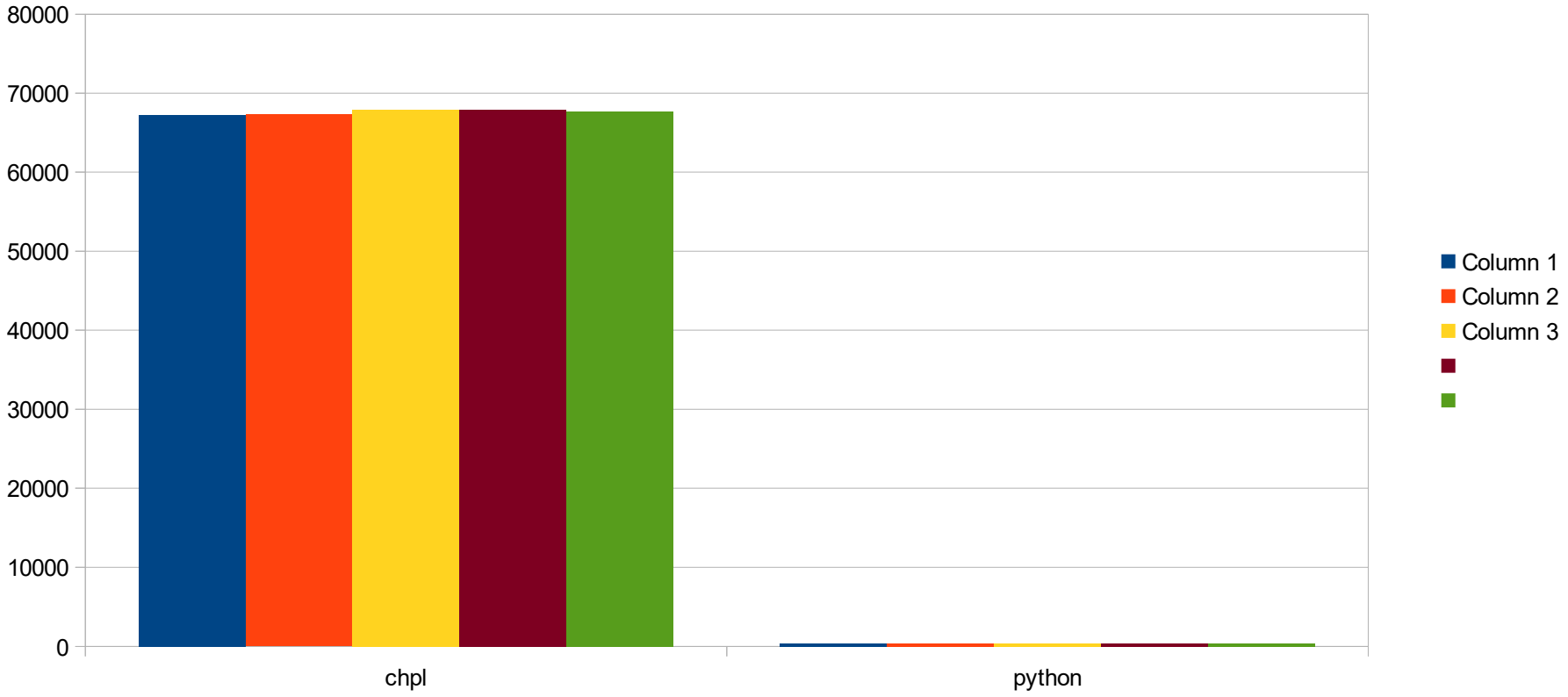


Theta Sketch

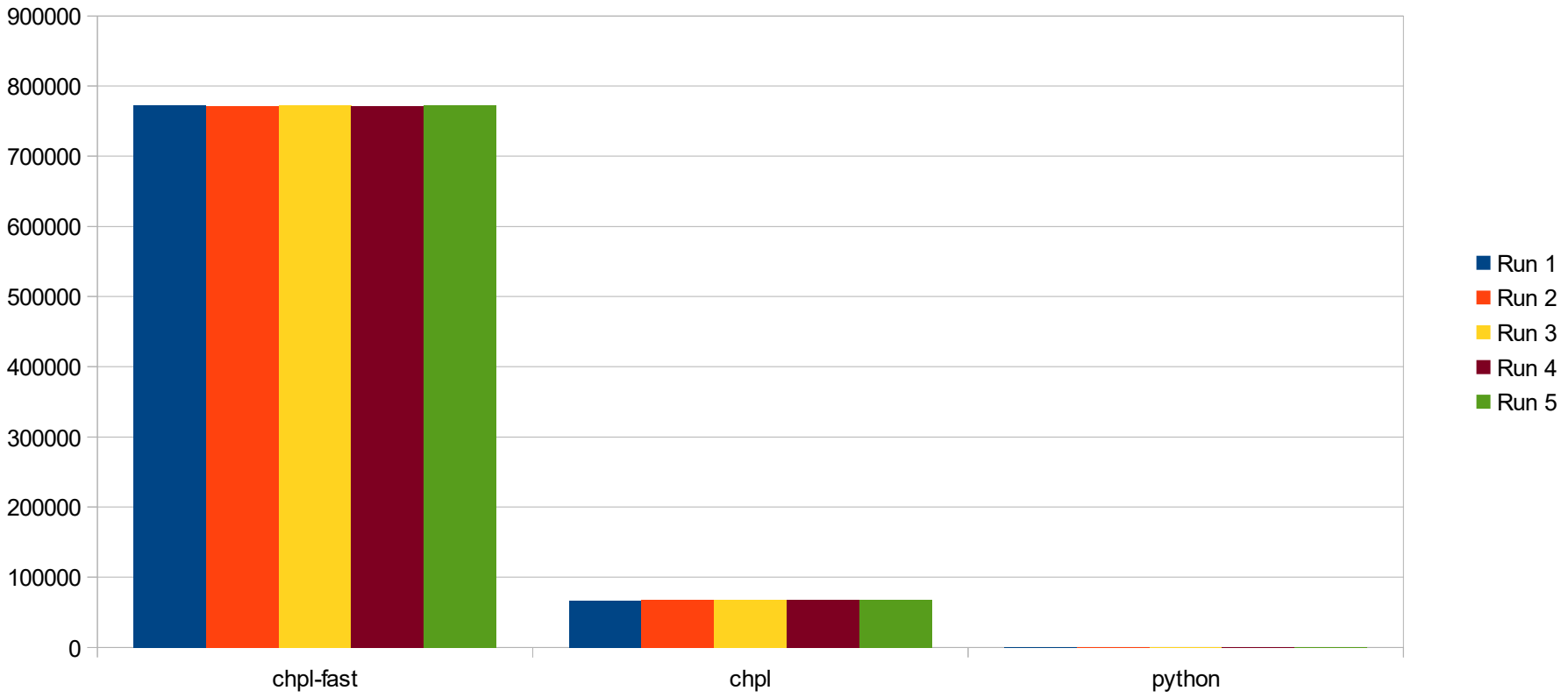
Theta Sketch

- Kth Minimum Value sketch
- Maintains a threshold theta and a set of unique hashed items less than theta
  - Assume hashing function computes a uniform distribution
- Algorithm assumes hash function provides uniform distribution (over hash space).
- The assumption gives information about the average spacing between elements of the stream.
- Knowing the smallest value, and spacing, one can infer the total number of distinct values observed

Theta Sketch



Theta Sketch



- Images provided by Library of Congress
  - All photos have “no known restrictions on publication”
- Code to be posted on github!
  - Check the email listserv for details