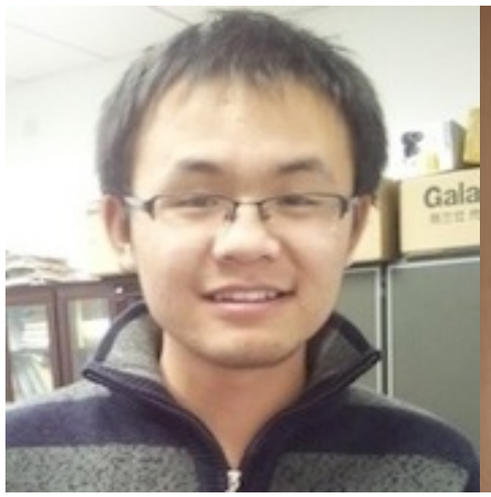
SlimFast

Reducing Metadata Redundancy in Sound & Complete Dynamic Data Race Detection

Joseph Devietti
University of Pennsylvania

collaborators



Yuanfeng Peng

Christian DeLozier

Ariel Eizenberg

William Mansky

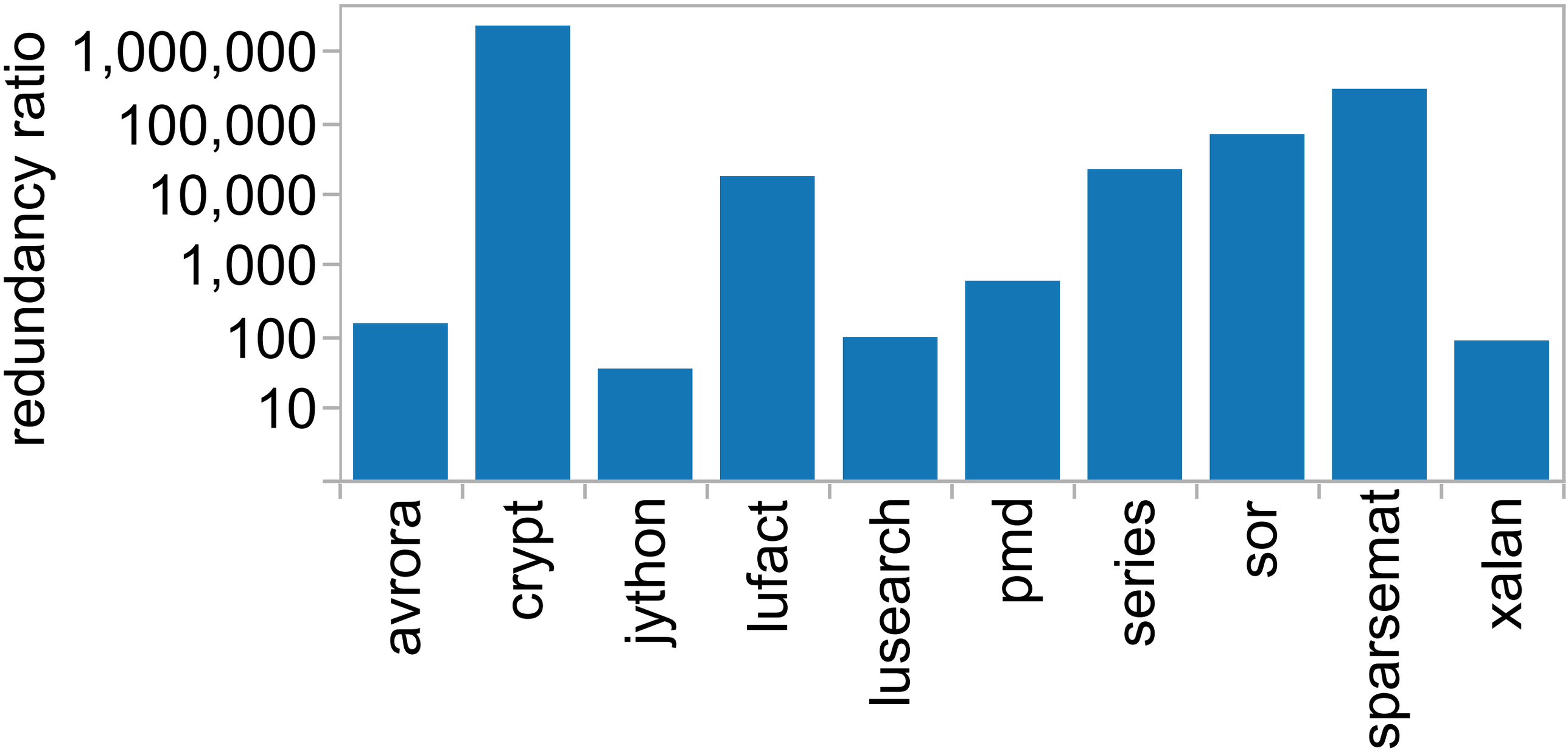
data race detection is useful

many uses **require**
sound+complete detection

data race detection is too slow

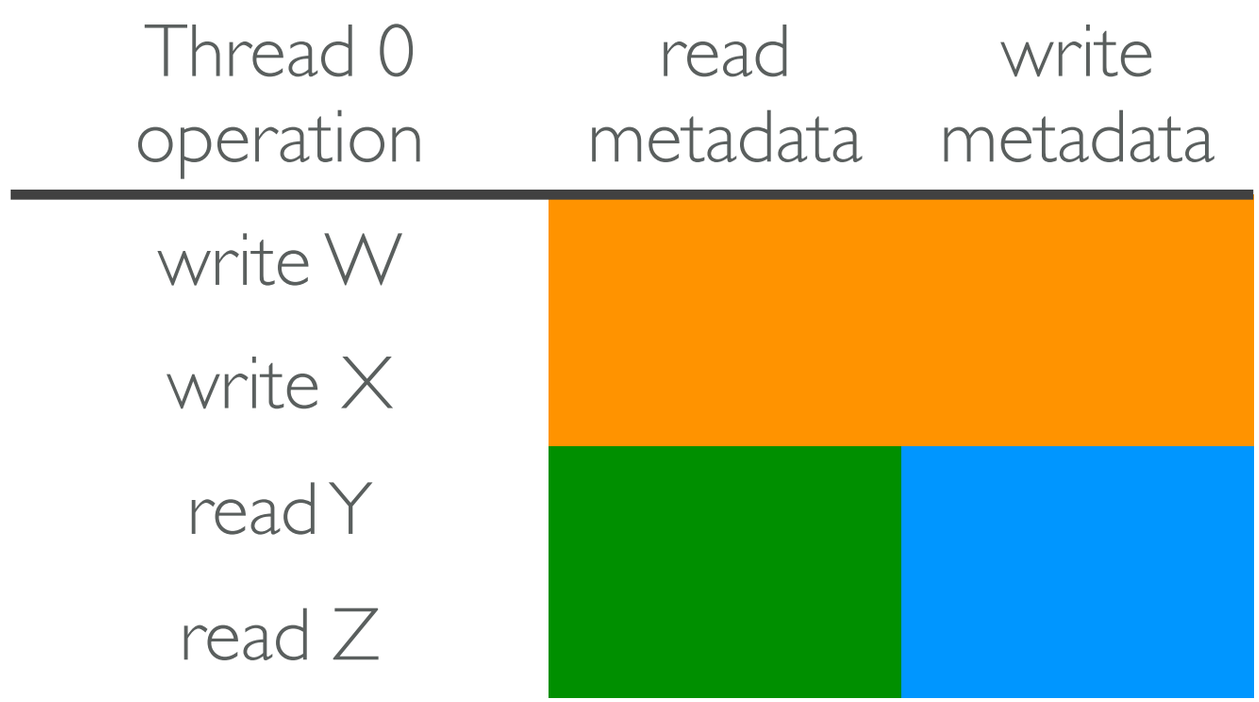
- **10x slowdown** over native execution with FastTrack [Flanagan & Freund, PLDI 2009, CACM Research Highlight 2010]
- **≥24 bytes of metadata** per program variable/array element
- lots of work to check for races

most metadata is redundant



why does redundancy arise?

memory accesses
without intervening synchronization
generate redundancy



SlimFast: Reducing Metadata Redundancy

- eliminate redundancy by storing only unique, immutable metadata
  - share these instances via pointers, a lá hash consing
- each thread maintains a **local set** of metadata it has allocated
- fully sound and complete

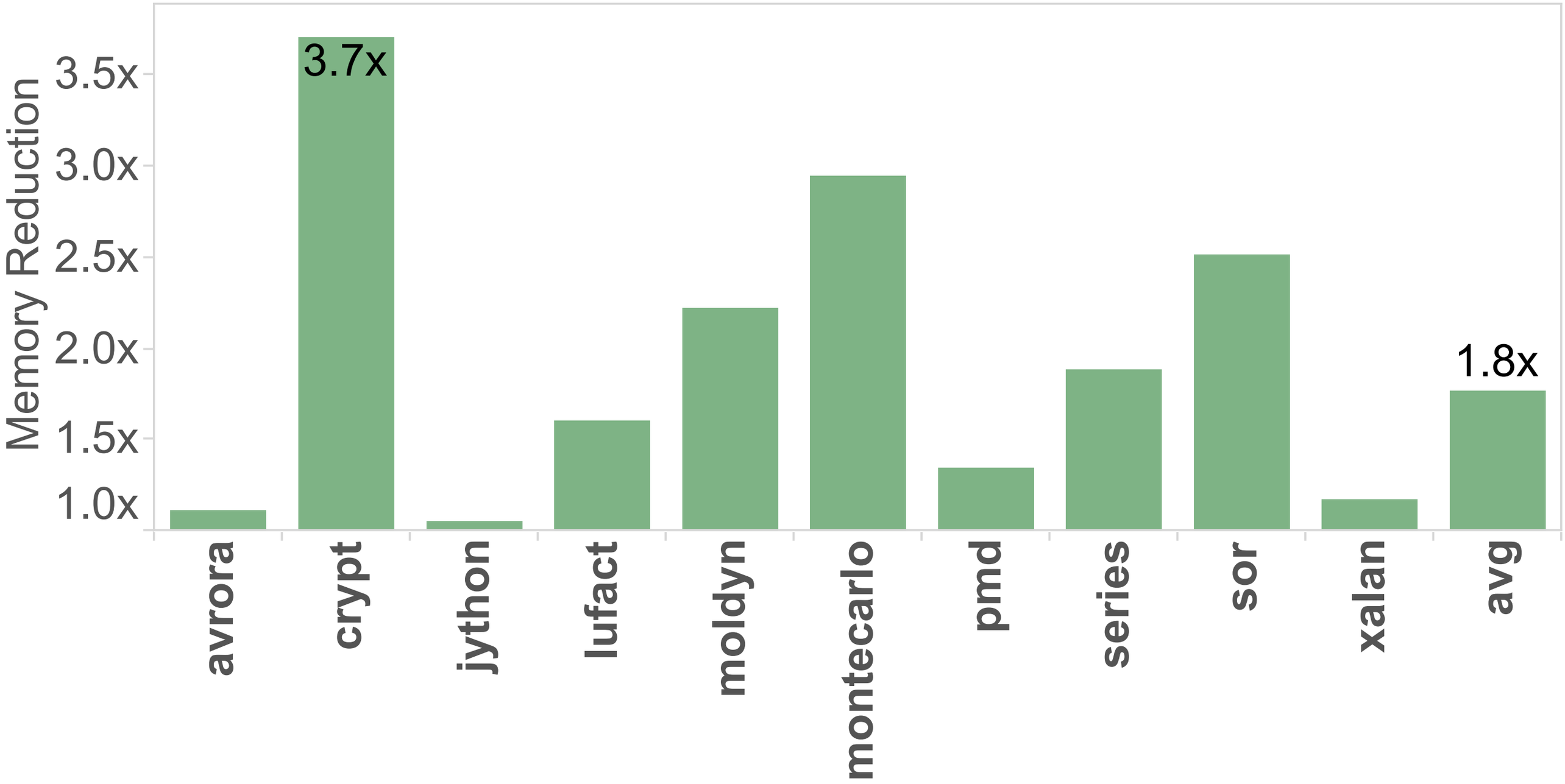
SlimFast optimizations

- local sets are **very small**, often with <5 elements
  - implement local sets with arrays instead of hash tables
- local sets provide **optimal redundancy elimination**
- immutable metadata enables **lock-free techniques**

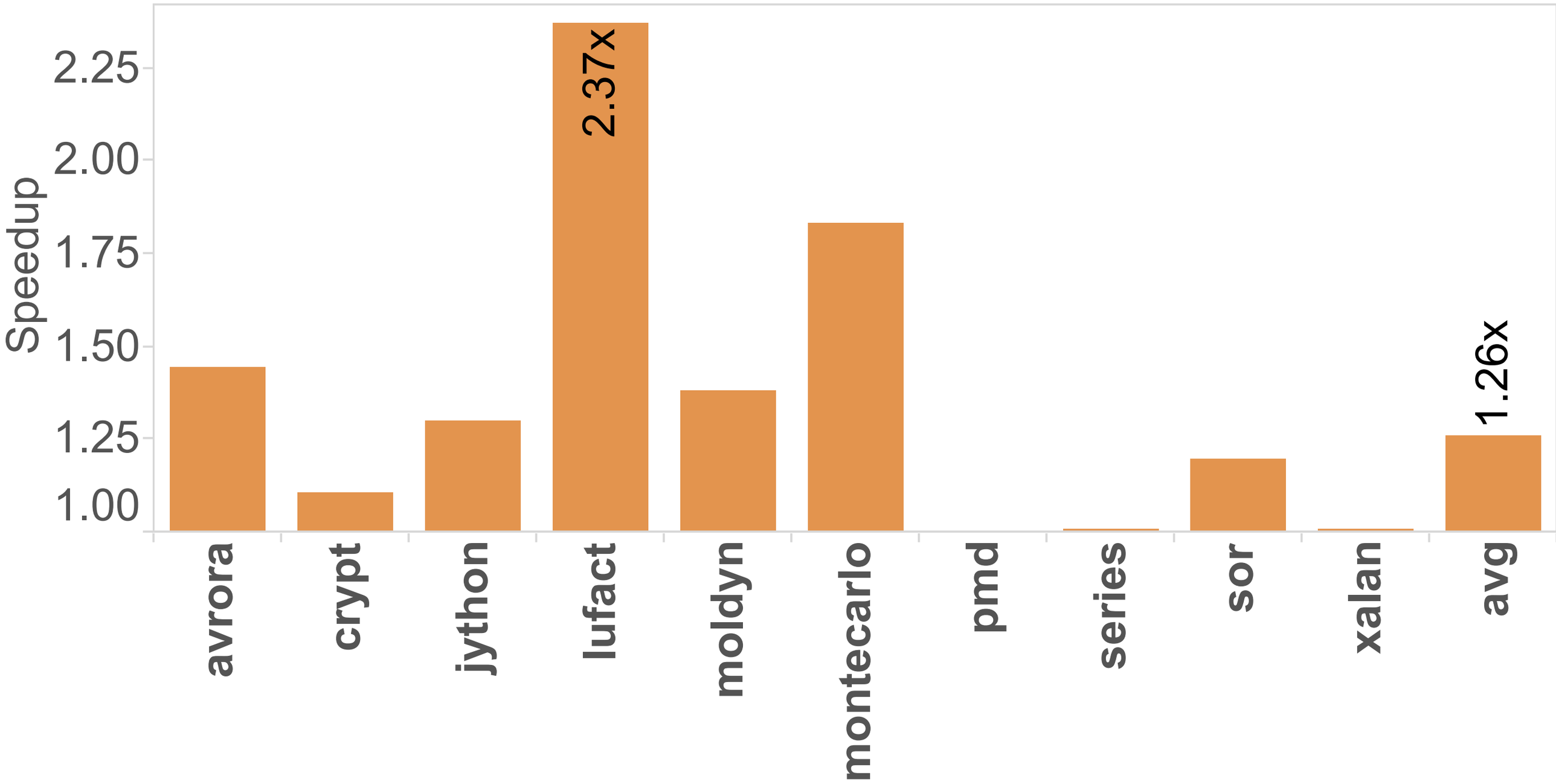
SlimFast prototype

- built in RoadRunner dynamic binary instrumentation framework for Java [Flanagan & Freund, PASTE 2010]
- compare with FastTrack data race detector
- use DaCapo and Java Grande benchmarks
- run with unconstrained memory (64GB RAM) to eliminate GC overheads, OS paging

SlimFast memory reduction



SlimFast speedup



future work

- apply these ideas to other concurrency analyses
  - atomicity violation detection
  - data race detection in structured parallelism
- explore other redundancies
  - among threads in SIMD/GPU programs