Fluid Co-Processing: GPU Bloom Filters for CPU Joins

Tim Gubner[*] Diego Tomé[*]
Harald Lang[^] Peter Boncz[*]

[*] CWI Database Architectures

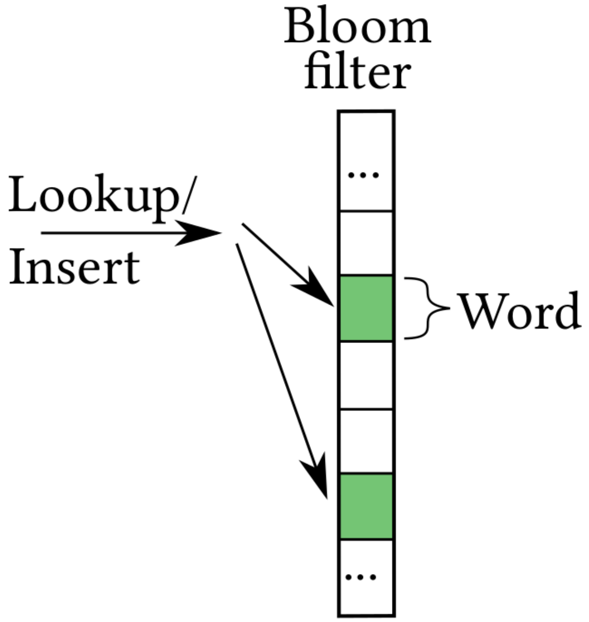
[^] TUM

Challenges in Co-Processing

- *What* can be offloaded?
  - DB algorithms rather data-intensive (shipping overhead)
  - Many do not reduce the amount of data
- *How* to offload?
  - Which device operates on what data?
  - Materialize table and send to GPU?
  - Send table chunks?
  - Due to time limitation: →Paper

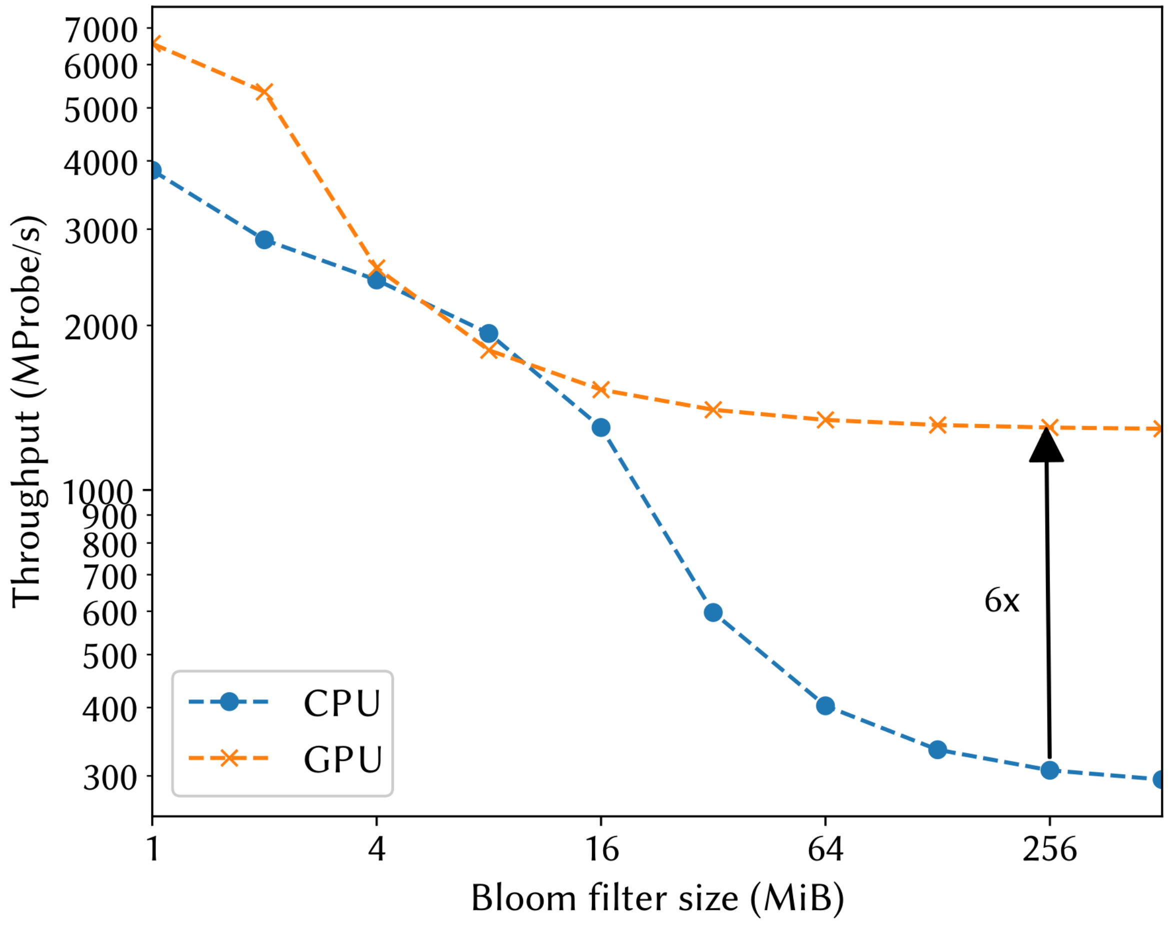
Bloom Filters*

- Tells whether item is *not* in set
- Insert:
  - Hash item using *k* hash functions
  - Set bits in bitset
- Lookup:
  - Hash using *k* hash functions
  - If all bits are 1 → might be in set
  - Otherwise → definitely not in set
- Useful for selective joins



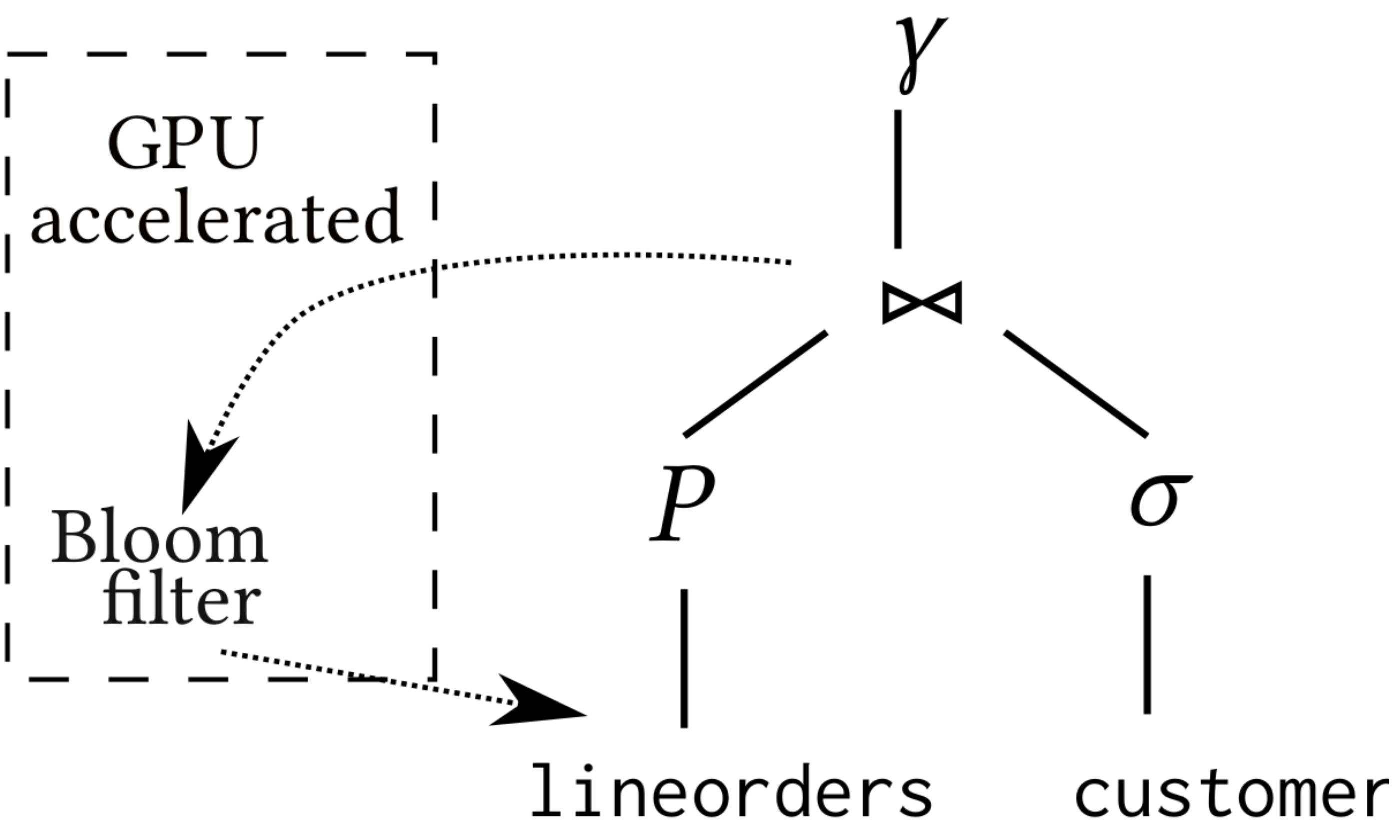
* H. Bloom: Space/Time Trade-offs in Hash Coding with Allowable Errors

Why Bloom Filters on GPU?

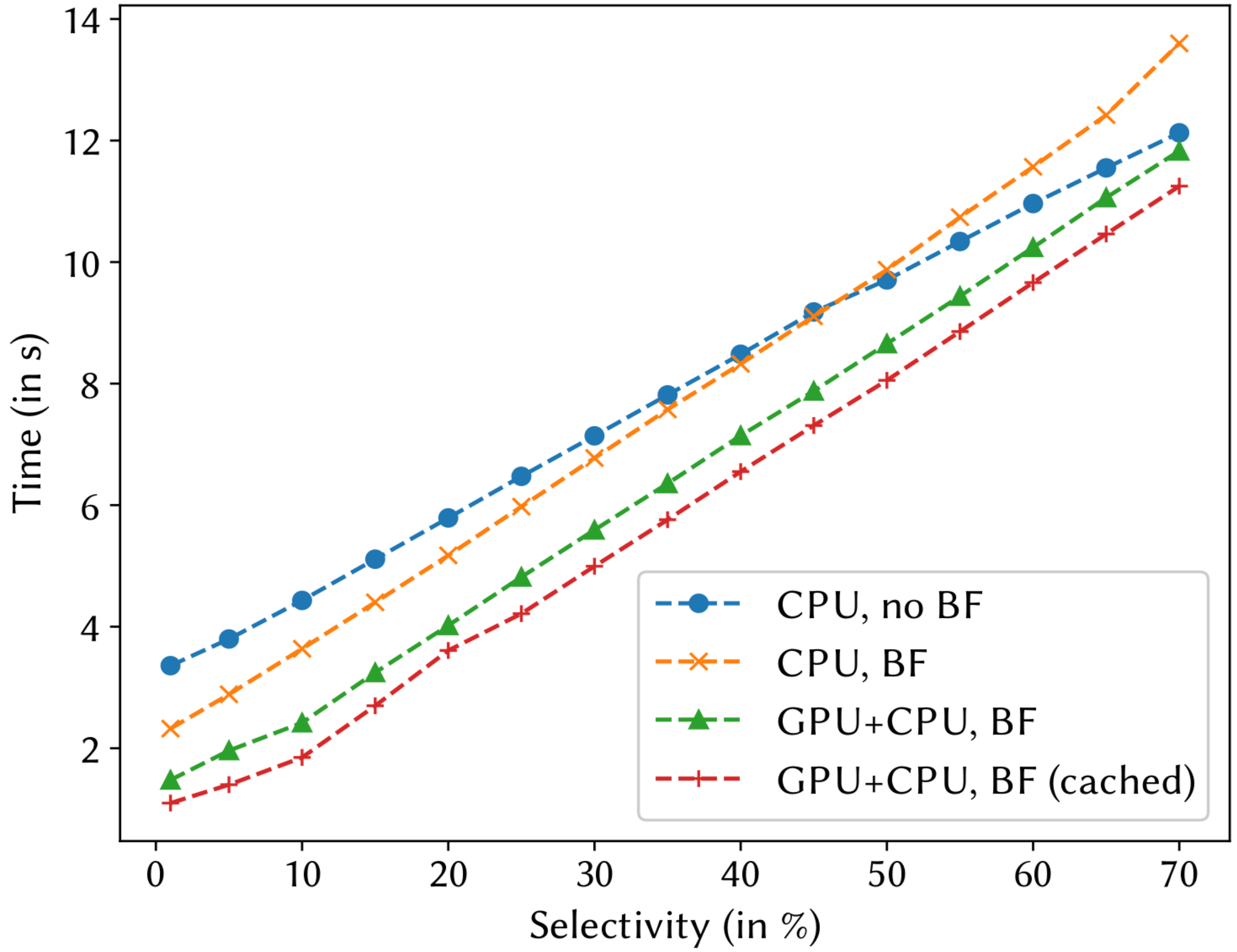


- Higher memory bandwidth (larger filters)
- Higher computational power (more hash functions)

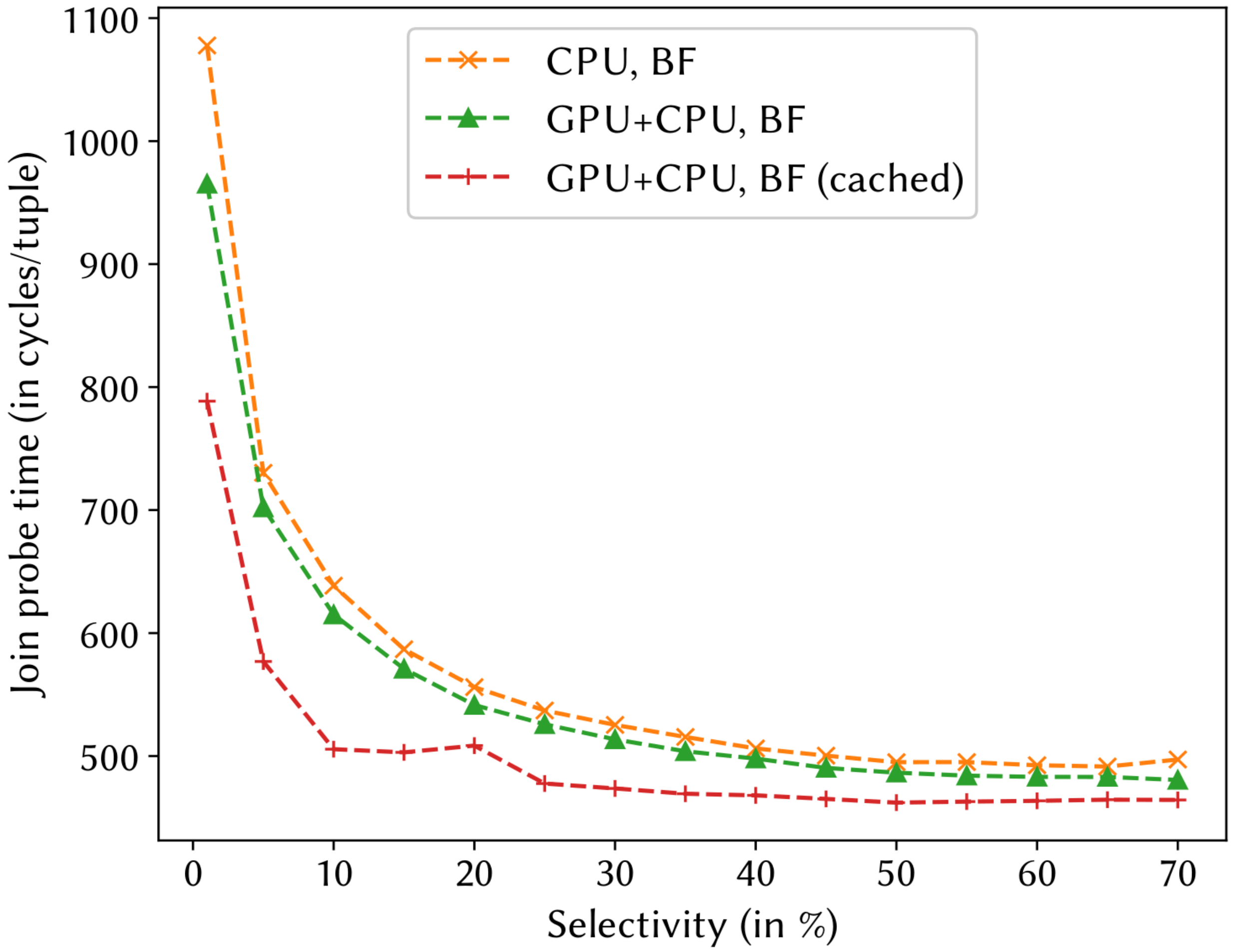
Case Study: Parallel Hash Join



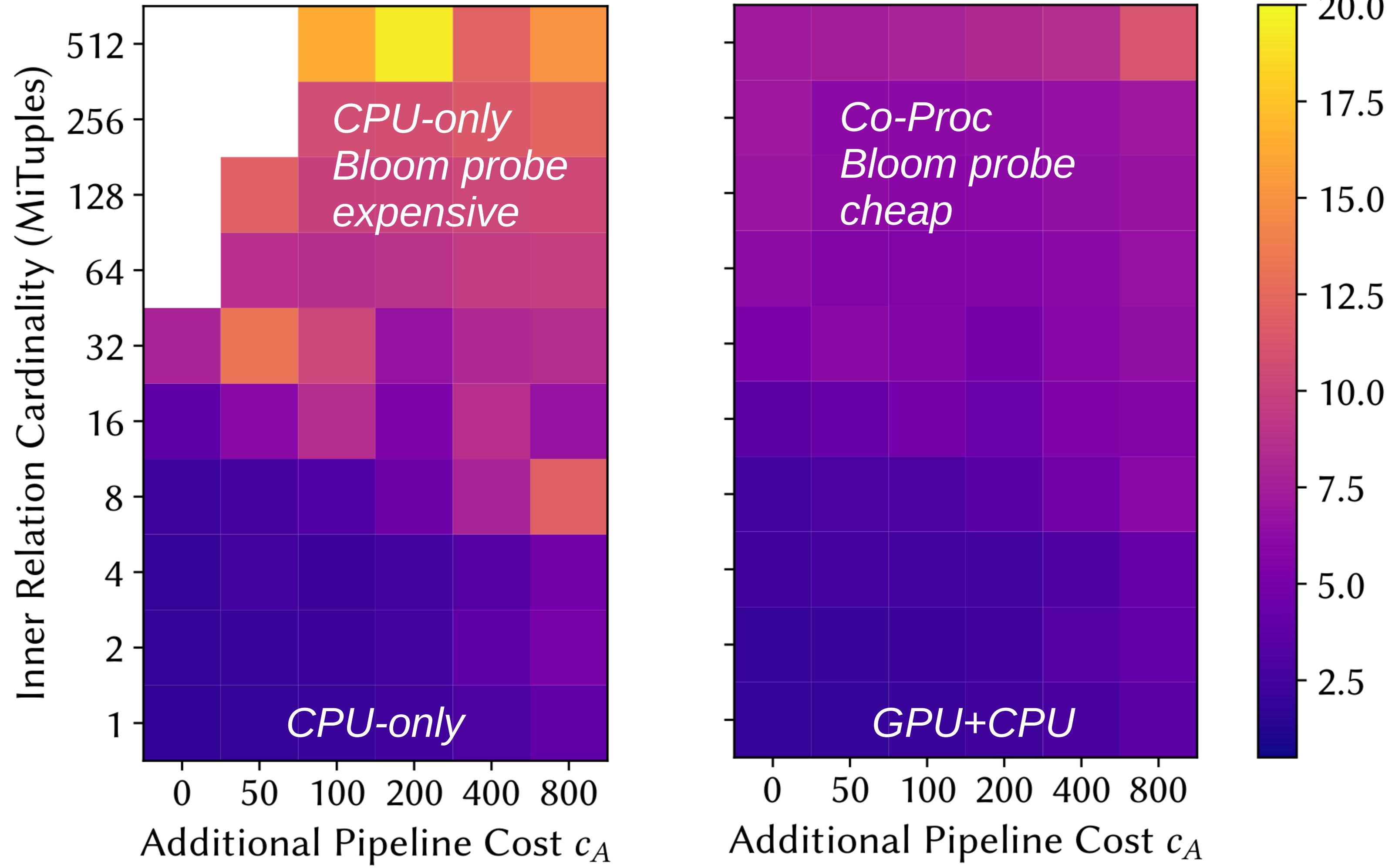
Lower Total Runtime



Faster Joins through Filter Offloading



Larger & faster Bloom Filters on GPU



Summary

- Offloading Bloom filter probing makes sense:
  - Not too much data movement
  - 6x faster on GPU
  - Allows large and precise Bloom filters (for large inner relations)
- Overall 3x faster
- For *fluid-co processing* and more details → Paper

Database Architectures