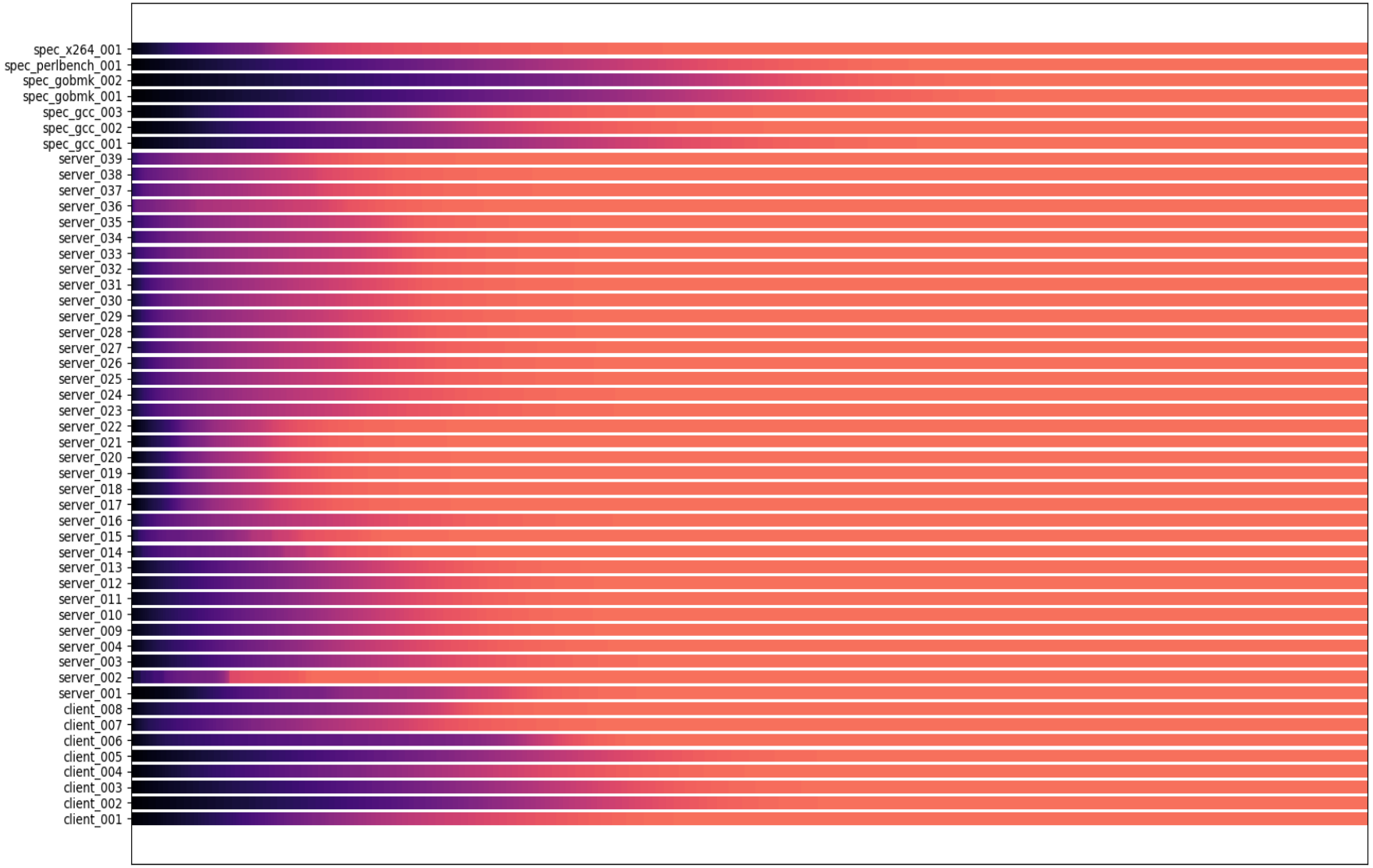
Temporal Ancestry Prefetching

Nathan Gober, Gino Chacon,
Daniel Jiménez, Paul V. Gratz
Texas A&M University

Block Dead Times in L1I

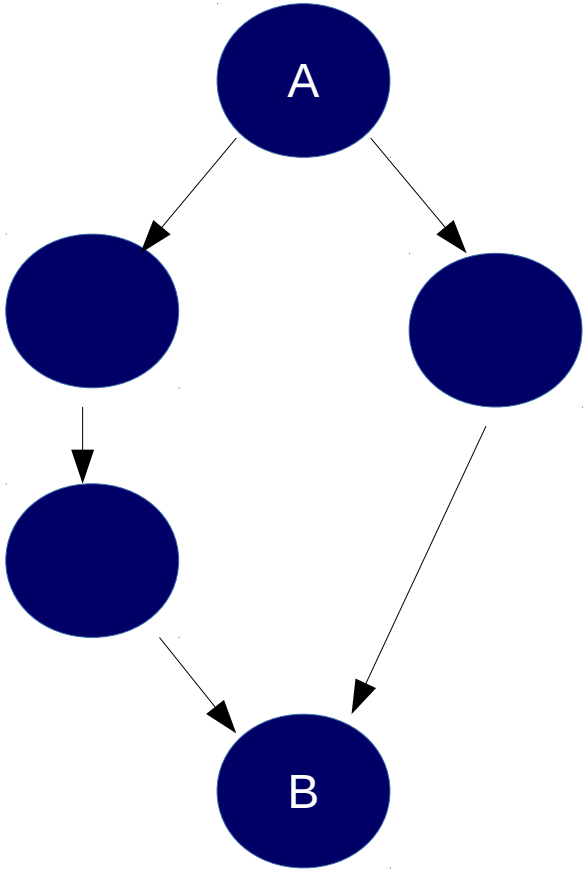


Outline

- Background
  - Ancestry Prediction
  - Temporal Prefetching
- Design
- Results
- Conclusion

Ancestry prediction

- Many paths of varying lengths from A to B
- Blocks in I-cache are long-lived
- Short histories miss the connection between A and B



Temporal Prefetching

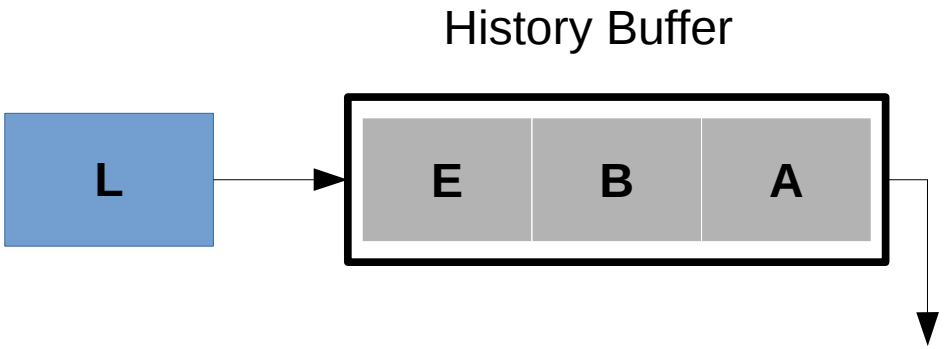
- Replay old misses on cache access
- Large metadata requirements
  - State of the art uses off-chip memory [Wenisch HPCA'09, Bakhshalipour HPCA'18]
  - Structural address space? [Wu ISCA'19]
- IPC1 gives large hardware budget

Outline

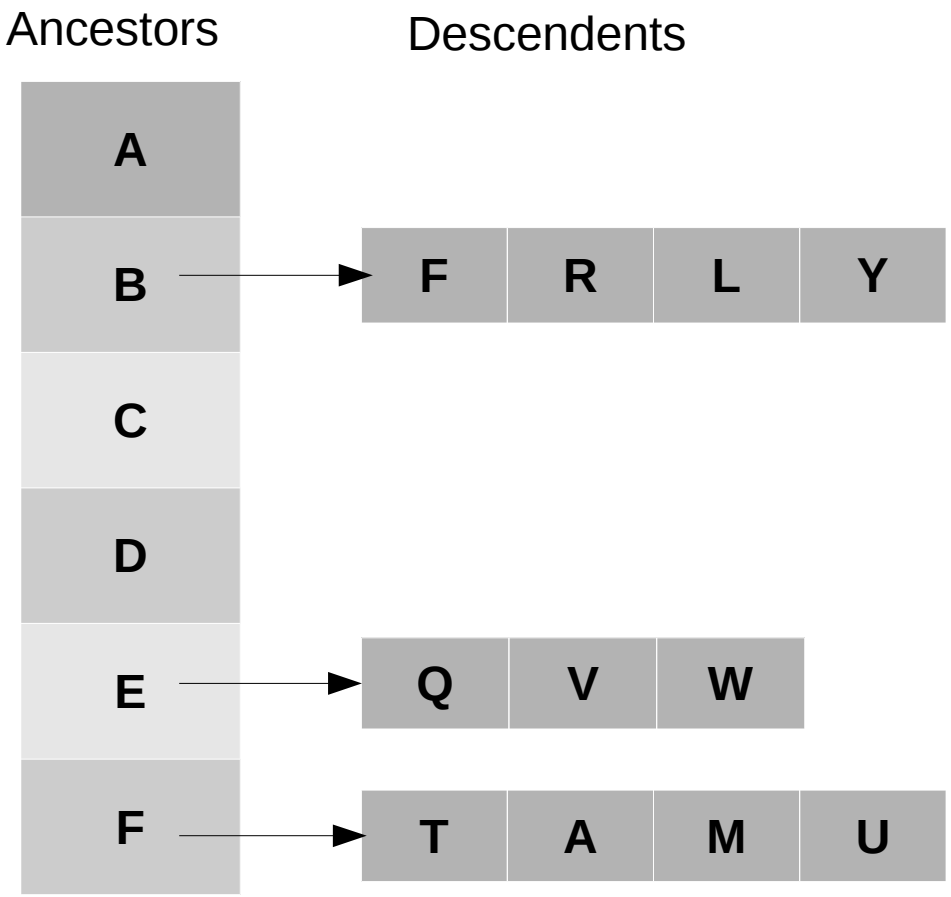
- Background
  - Ancestry Prediction
  - Temporal Prefetching
- Design
- Results
- Conclusion

TAP Model (Training)

- Maintain path history
- De-duplicated
- A single PC has many histories

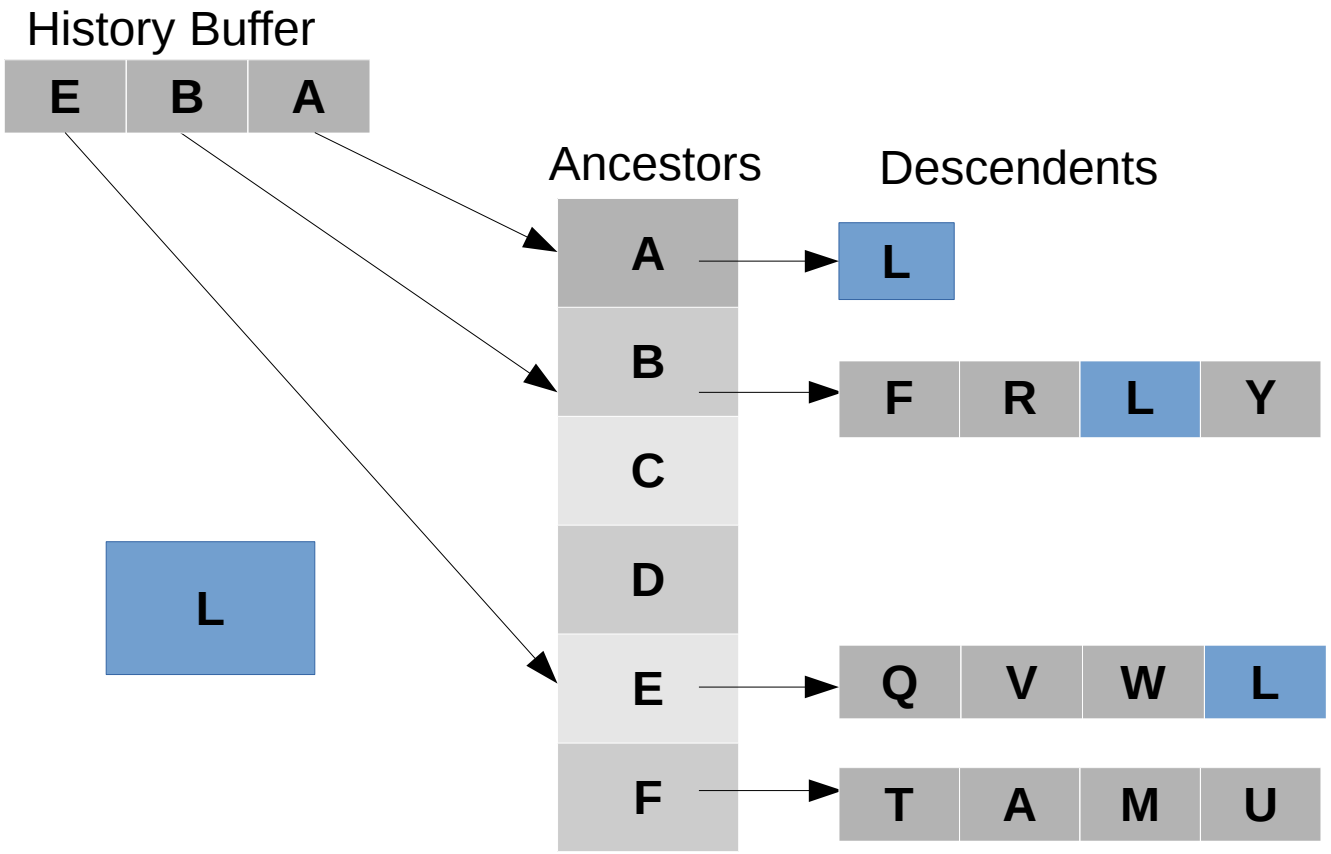


TAP Model (Training)



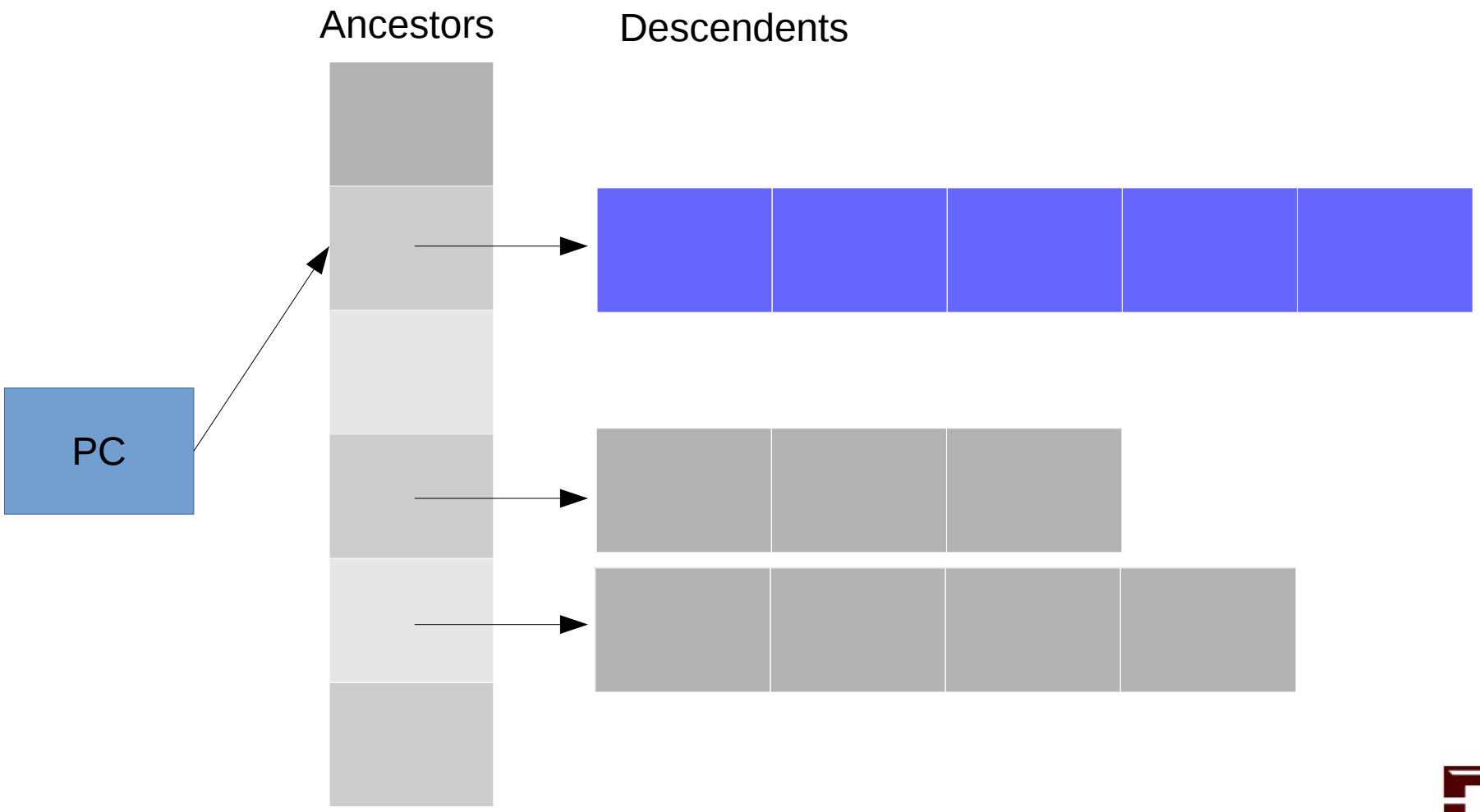
- Descendents are captured misses
- The path is not recorded, only the eventual predicted miss

TAP Model (Training)



- Increment weights on cache access
- Decrement weights on eviction if prefetch not useful
  - Invalidate instead if weight was zero.

TAP Model (Prediction)

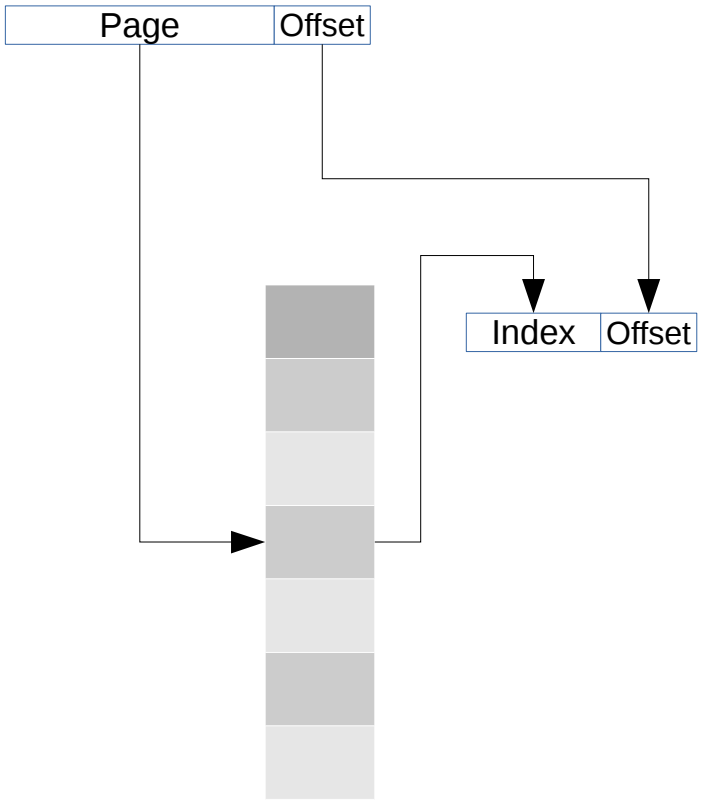


Prefetch Filtering

- Long descendency lists produce large numbers of prefetch hits.
- Filter prefetches by tag check.
- With shadow cache of 12-bit partial tags, filtering is 99% accurate.

Page Compression

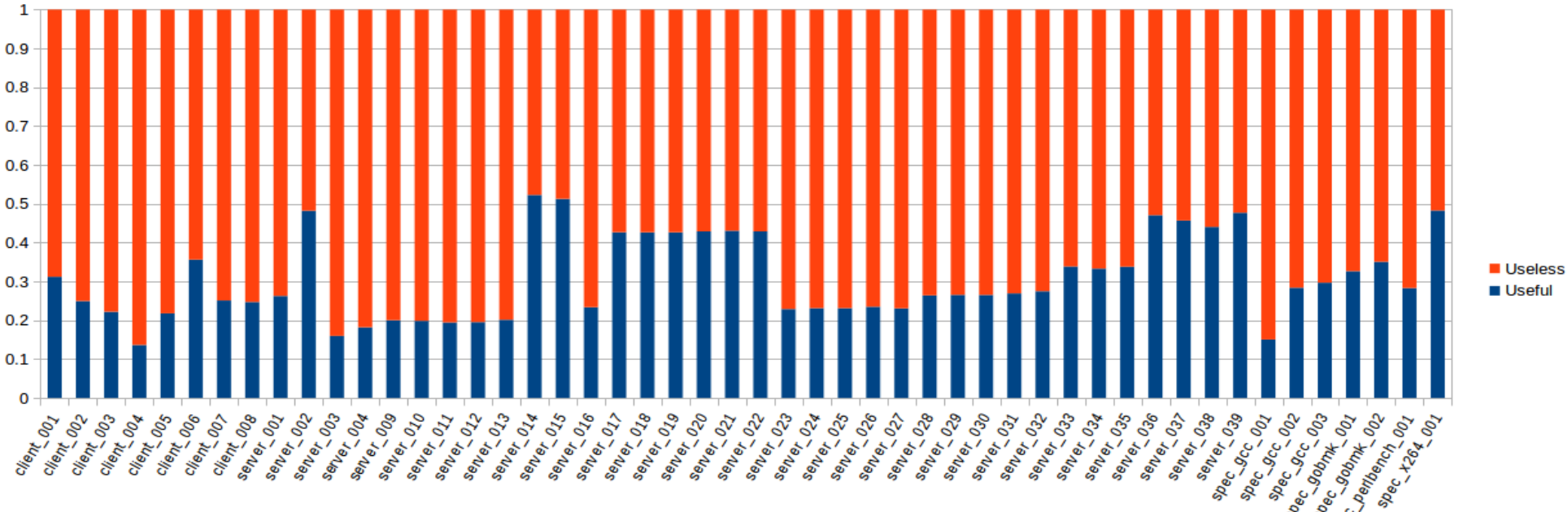
- Much metadata space is wasted on storing a few pages.
- Use buffer index as a proxy for page number.
- NRU replacement



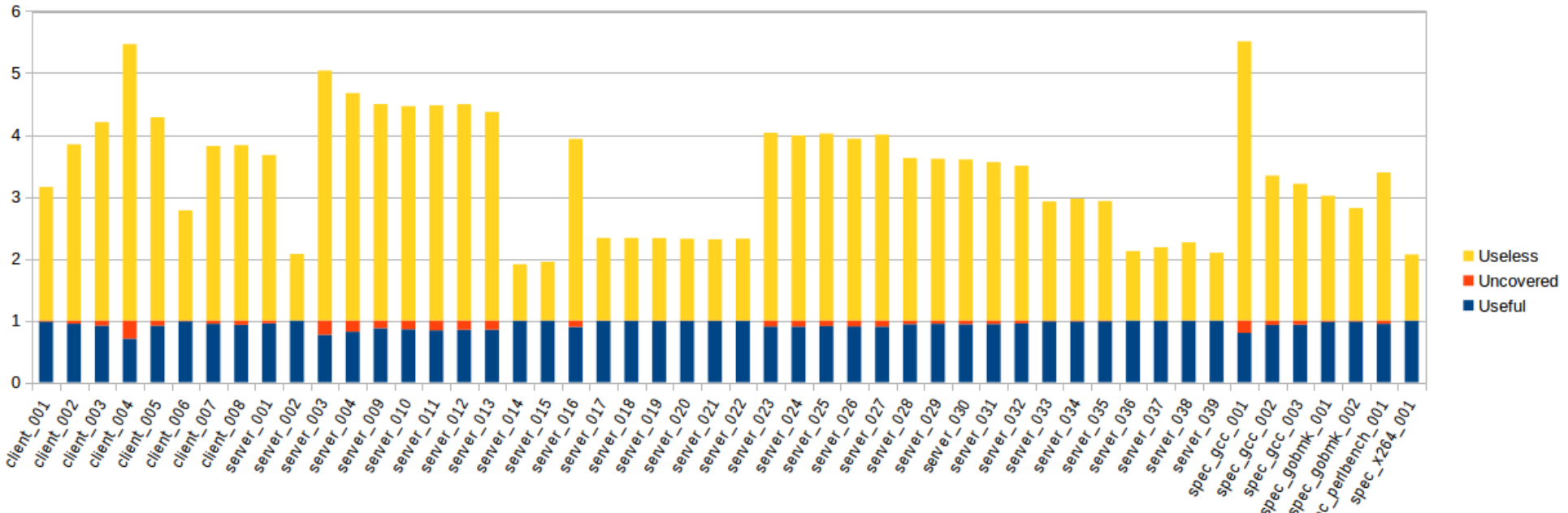
Outline

- Background
  - Ancestry Prediction
  - Temporal Prefetching
- Design
- Results
- Conclusion

Accuracy


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
client_001
client_002
client_003
client_004
client_005
client_006
client_007
client_008
server_001
server_002
server_003
server_004
server_009
server_010
server_011
server_012
server_013
server_014
server_015
server_016
server_017
server_018
server_019
server_020
server_021
server_022
server_023
server_024
server_025
server_026
server_027
server_028
server_029
server_030
server_031
server_032
server_033
server_034
server_035
server_036
server_037
server_038
server_039
spec_gcc_001
spec_gcc_002
spec_gcc_003
spec_gobmk_001
spec_gobmk_002
spec_perlbench_001
spec_x264_001
Useless
Useful

Coverage


6
5
4
3
2
1
0
Useless
Uncovered
Useful
client_001
client_002
client_003
client_004
client_005
client_006
client_007
client_008
sever_001
server_002
server_003
server_004
sever_009
server_010
server_011
server_012
server_013
server_014
server_015
server_016
server_017
server_018
server_019
server_020
server_021
server_022
server_023
server_024
server_025
server_026
server_027
server_028
server_029
server_030
server_031
server_032
server_033
server_034
server_035
server_036
server_037
server_038
server_039
spec_gcc_001
spec_gcc_002
spec_gcc_003
spec_gobmk_001
spec_gobmk_002
spec_perlbench_001
spec_x264_001

Conclusion

- Tracking cache evictions is useful
- Prefetch sequencing is unimportant
- Future Work
  - Replacement in L1I
  - Stricter throttling on poor-performing workloads