Pangloss: a novel Markov chain prefetcher

The 3rd Data Prefetching Championship (co-located with ISCA 2019)

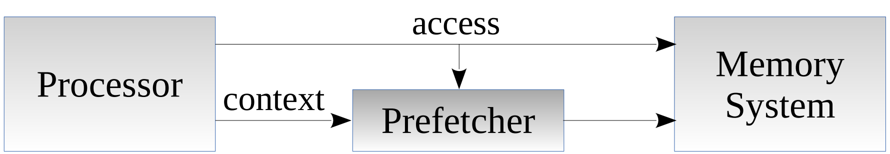
Philippos Papaphilippou, Paul H. J. Kelly, Wayne Luk

Department of Computing, Imperial College London, UK

{pp616, p.kelly, w.luk}@imperial.ac.uk

Data Prefetchers

- The task:
  - Predict forthcoming access addresses
  - Hardware mechanism → Agnostic to workload
    - Space and logic limitations
    - Software alternatives exist



- Multiple approaches for predicting the most likely next accesses
  - Through the address stream that was already-seen
    - Repeating sections
    - Repeating sections relative to the page
    - Delta transitions
  - Context-based, such as with correlating with
    - Page
    - Instruction Pointer (IP)
    - CPU Cycles
- Other concerns: Throttling mechanisms, most profitable predictions, energy

Distance Prefetching

- A generalisation of Markov Prefetching
  - Originally: model address transitions
  - Approximate a Markov chain, but
  - Based on Deltas instead of Addresses

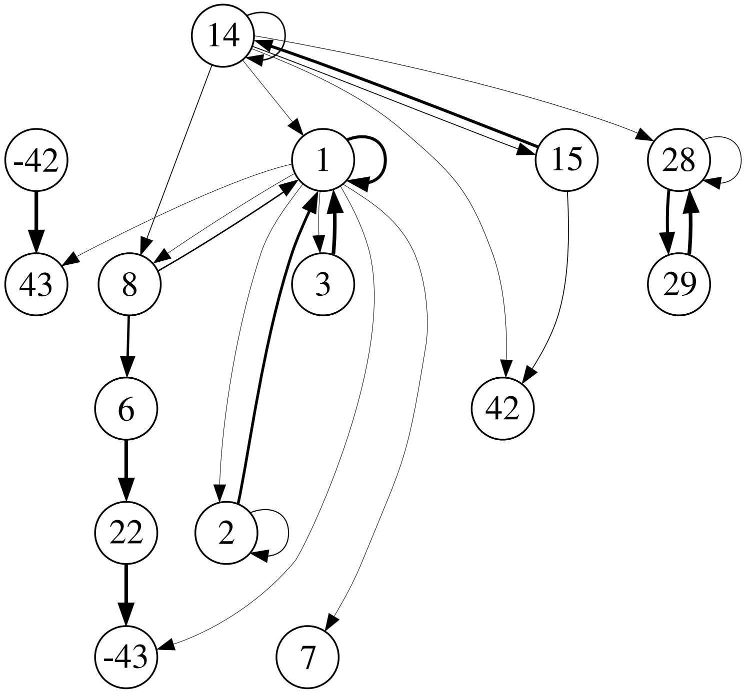
    $Delta = Address - Address_{Prev}$

    - Use the model to prefetch the most probable deltas

      $Address_{Next} = Address + Delta_{Next}$
- Deltas example

```
Address:      1    4    2    7    8    9
Delta:             3   -2    5    1    1
```

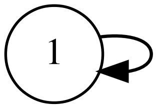
- Delta transitions
  - More general than address transitions
    - Different addresses
  - Can be meaningful to use globally
    - Different pages, IPs, etc.



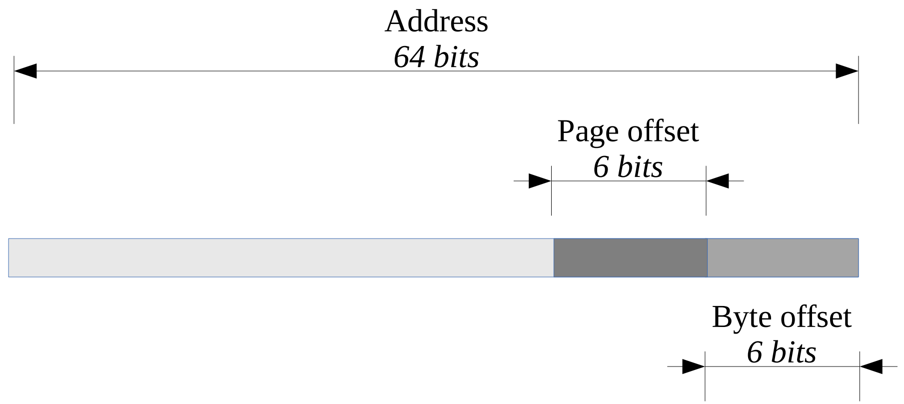
Markov Model (cactuBSSN)

Prefetching in the framework (ChampSim)

- Providing one prefetcher for each of the L1, L2 and Last-Level Cache (LLC)
- Last address bits (L2)
  - Cache line (byte) offset: 6-bits → Representing $2^6 = 64$ bytes
  - Page (byte) offset: 6-bits → Representing $2^{6+6} = 4$K bytes
- Address granularity
  - L1: 64-bit words → 512 positions in a page
  - L2: cache line → 64 positions in a page
  - L3: cache line → 64 positions in a page
- Distance prefetching is limited by the page size
  - Page allocation/translation is considered random
  - Unsafe/unwise to prefetch outside the boundaries
- Example in L2 for delta transition (1, 1)




```
..1010011010111100XXXXXX    saw
..1010011010111101XXXXXX    saw
..1010011010111110XXXXXX    saw
..1010011010111111XXXXXX    prefetch
..1010011011000000XXXXXX    ~~prefetch~~ discard
```

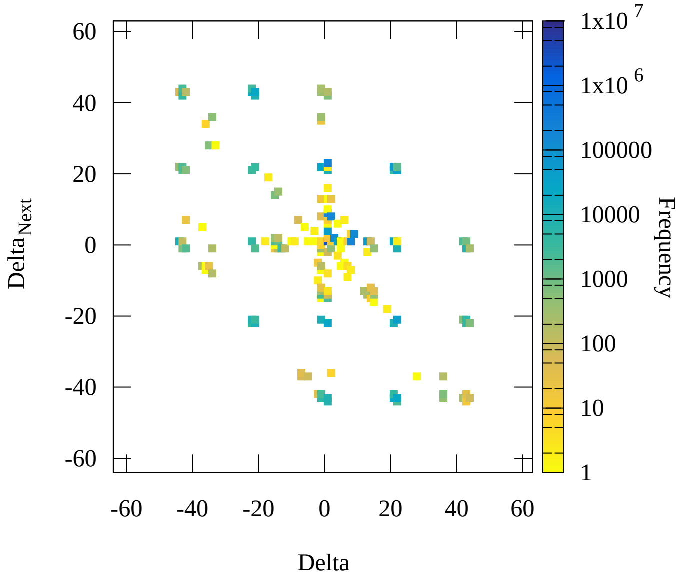


Preliminary experiment

- Gain insights for
  - Optimisation
  - Understanding complexity of access patterns
- 46 benchmark traces
  - Based on the provided set of SPEC CPU2017, for which *MPKI > 1*
- Produce an adjacency matrix for delta transition frequencies

```
On Access:
    If on the same page:
        A[Delta_prev][Delta] += 1
```

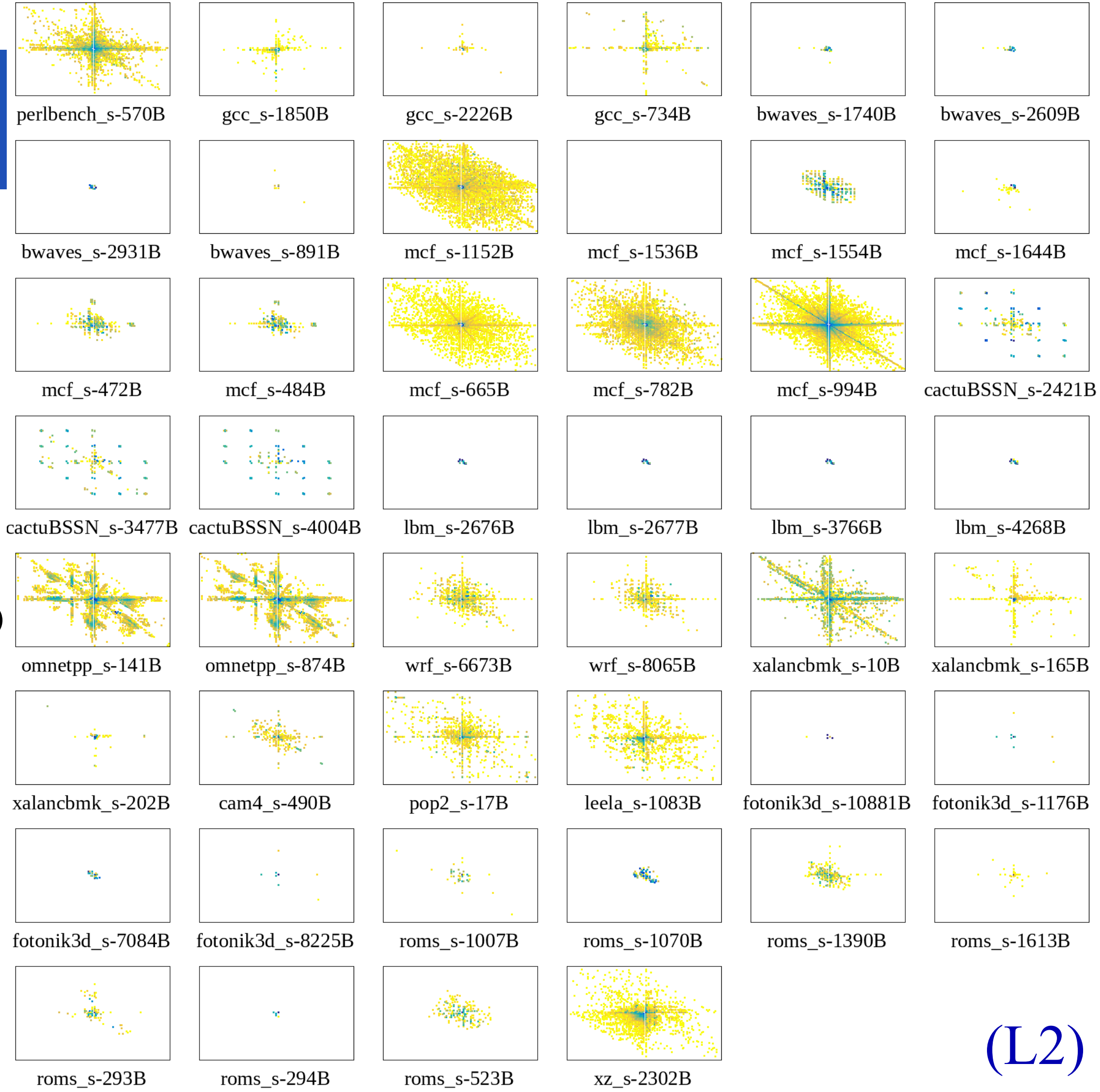
- Dummy prefetchers (only observing) for
  - L1D
  - L2
  - LLC



Adjacency Matrix (cactuBSSN)

Observations

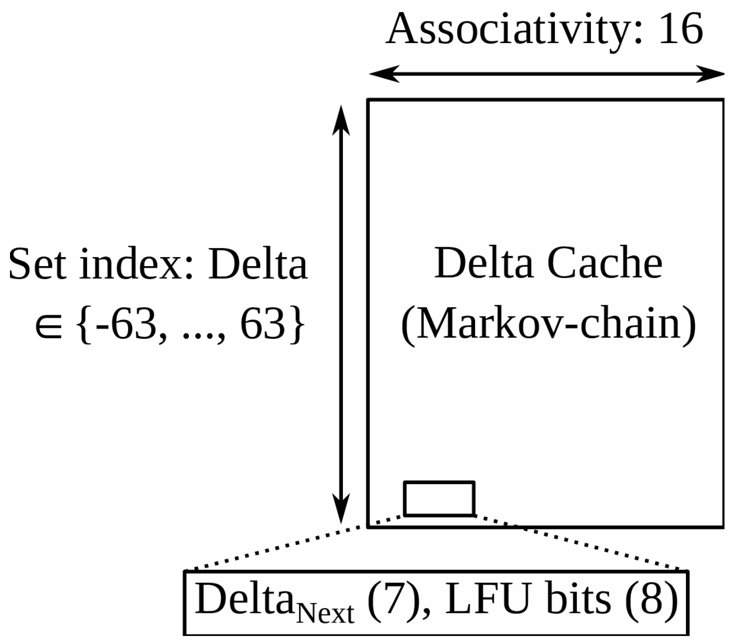
- Relatively sparse
  - No need for N×N matrix
- Complex access patterns
  - Simpler prefetchers might not be enough (e.g stride prefetching)
- Diagonal (& vertical/ horizontal) lines:
  - Random accesses when performing regular strides.
  - Example: (1,1) → (1, -40) → (-40, 41) → (41, 1) → (1,1)
  - Resulting in new lines: y=-x+1, x=1, y=1
- Hexagonal shape:
  - Such outliers would point outside the page
  - Example: (50, 50) totals to a delta of 100 ≥ 64
- Sparse or empty matrices: (see mcf_s-1536B)
  - Simple patterns or
  - Many invalidated deltas



(L2)

Key idea: H/W representation with increased accuracy

- Related work
  - Markov chain stored in associative structures
    - Set-associative
    - Fully-associative → expensive
  - No real metric of transition probability
    - Using common cache replacement policies → based on recency
      - First Come, First Served (FCFS)
      - Least Recently Used (LRU)
      - Not-Most Recently Used (NRU)
- Our approach
  - Set-associative cache
    - Indexed by previous delta
  - Pointing to next most probable delta
  - (Least Frequently Used) LFU-inspired replacement policy
    - On hit, the counter in the block is incremented by 1
    - On a counter overflow, divide all counters in the set by 2
      → maintaining the correct probabilities



Markov Chain in H/W

Invalidated deltas

- Interleaving pages can ‘hide’ valid deltas
  - $Delta = Address - Address_{Prev.}$ is not enough
- Example
  - 1010011010111100XXXXXX
  - 0101100101000100XXXXXX
  - 1010011010111101XXXXXX
  - 0101100101000111XXXXXX

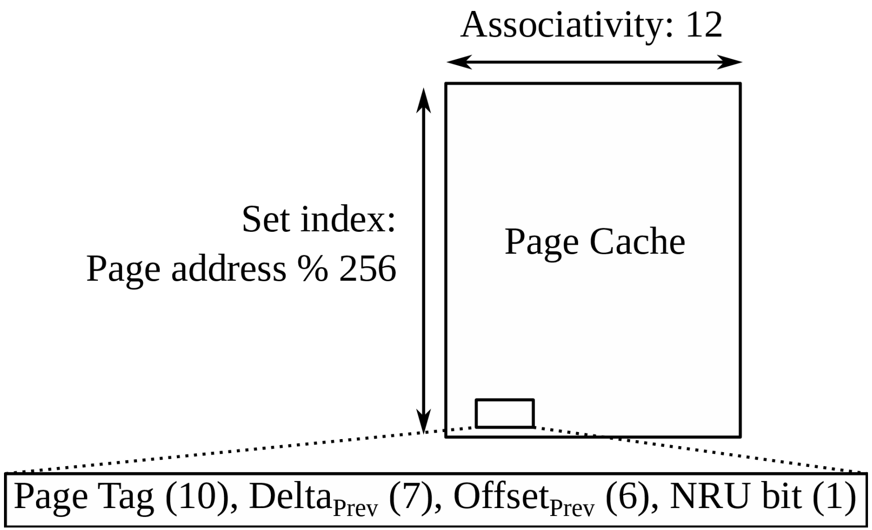
  +1

  +3
- Common cases
  - Out-of-order execution in modern processors
  - Reading from multiple sources iteratively
    - merge sort → multiple mergings of two (sub) arrays

Invalidated deltas solution

- *(small resemblance in related work, such as in VLDP [5], KPCP [6])*
- Track deltas and offsets per page
- Providing a H/W-friendly structure
  - Set-associative cache
  - Indexed by the page
  - Holding last delta and offset per page
    - Also the page tag and the NRU bit
- Building delta transitions
  - If page match:

    $(Delta_{Prev}, Offset_{Prev} - Offset_{Curr})$
  - Update the Markov Chain

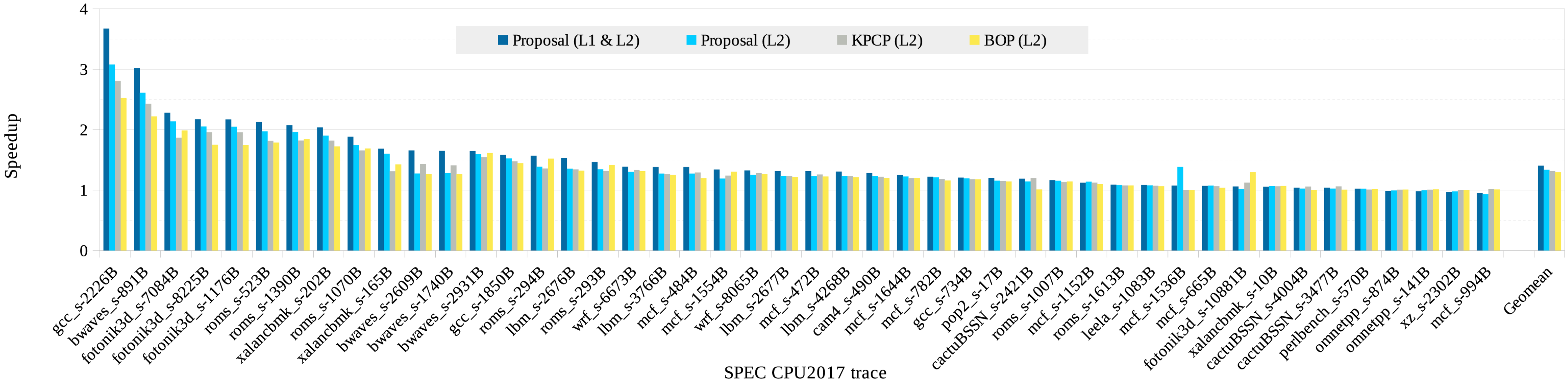


Per page information

Single-thread performance

- Pangloss (L1&L2) speedups: 6.8%, 8.4%, 40.4% over KPCP, BOP, non-prefetch
- For fairness we report the same metrics for our single-level (L2) version
  - 1.7% and 3.2% over KPCP and BOP.

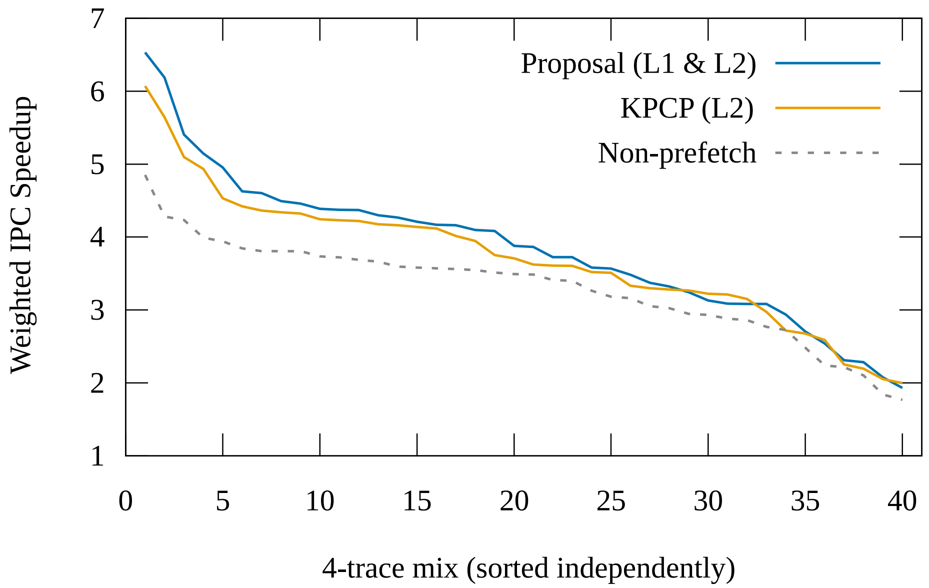
$$Geometric\ Speeup = \prod_{i=1}^{46} \frac{IPC_i^{prefetch}}{IPC_i^{non\ prefetch}}$$



Multi-core performance

- Producing 40 4-core mixes from the 46 benchmark traces
  - First, classify the traces according to their speedup from Pangloss (1-core)
    - Low: speedup ≤ 1.3
    - High: speedup > 1.3
  - Produce 8 random mixes for each of the following 5 class combinations
    - Low-Low-Low-Low (4 low)
    - Low-Low-Low-High (3 low & 1 high)
    - …
    - High-High-High-High (4 high)
- Evaluate using the weighted IPC speedup
  - 4-core speedup in each mix:

$$\sum_{i=1}^{4} \frac{IPC_i^{together}}{IPC_i^{alone, non\ prefetch}}$$



Hardware cost

- Space
  - Single-core: 59.4 KB total
    - (13.1 KB for single-level (L2))
  - Multi-core: 237.6 KB total

	Description (bits)	(KB)
L1D:		
Delta cache	*1024 sets × 16 ways × (10 + 7)*	*34.8*
Page cache	*256 sets × 12 ways × (10 + 10 + 9 + 1)*	*11.5*
L2:		
Delta cache	*128 sets × 16 ways × (7 + 8)*	*3.8*
Page cache	*256 sets × 12 ways × (10 + 7 + 6 + 1)*	*9.2*
LLC:	None	*0.0*
Total		*59.4*

TABLE I
SINGLE-CORE CONFIGURATION BUDGET

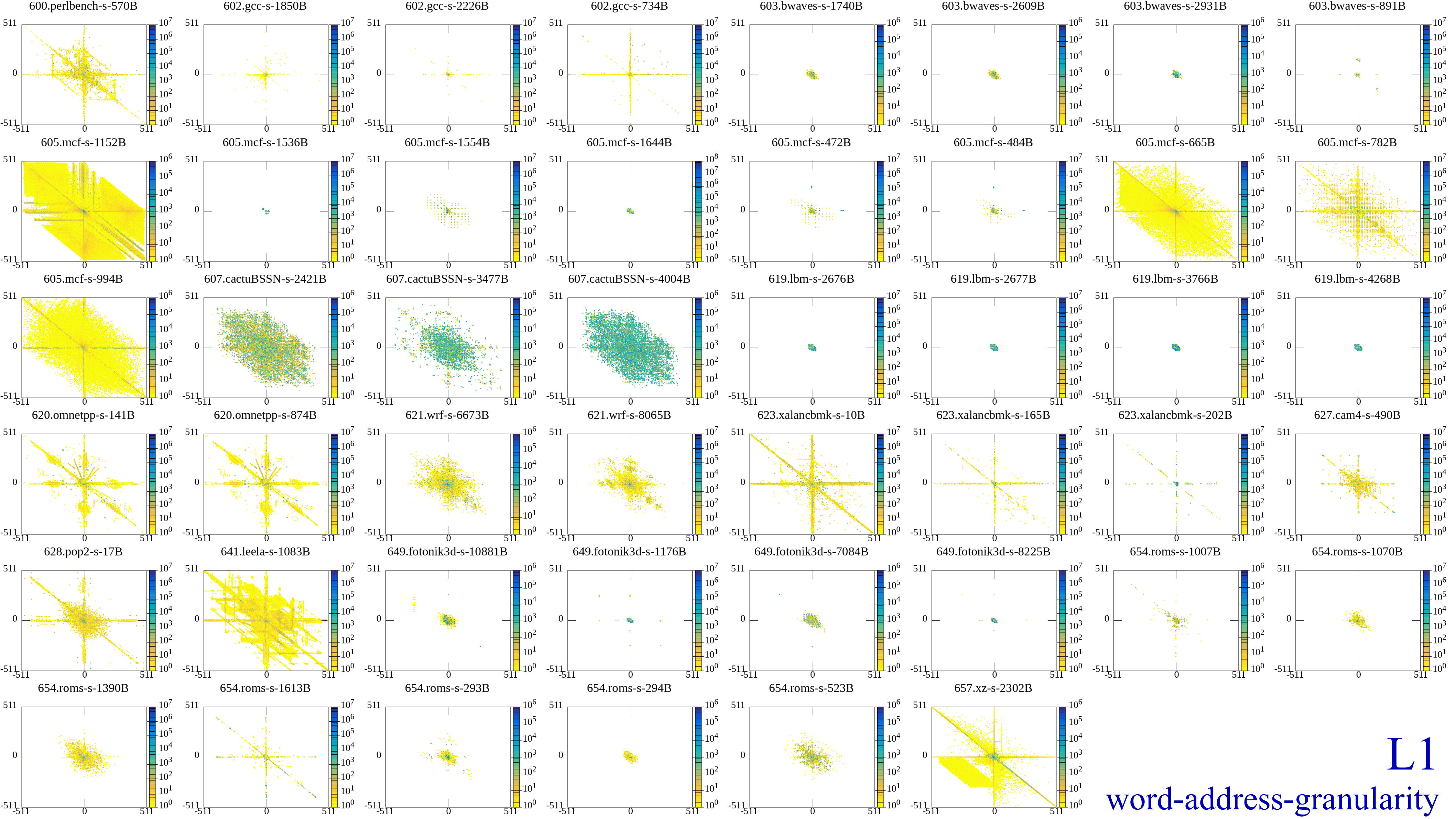
- Logic (insights)
  - Low associativity
    → up to 16 simultaneous comparisons
  - Traversal heuristic: select prob. > 1/3
    → no need to sort
    → only 2 candidate children per layer
  - Traversal heuristic: iterative
    → could be relatively expensive, but a delay could actually help with timeliness
  - IP and cycle information not used
- Can be fine-tuned according to the use case requirements

END

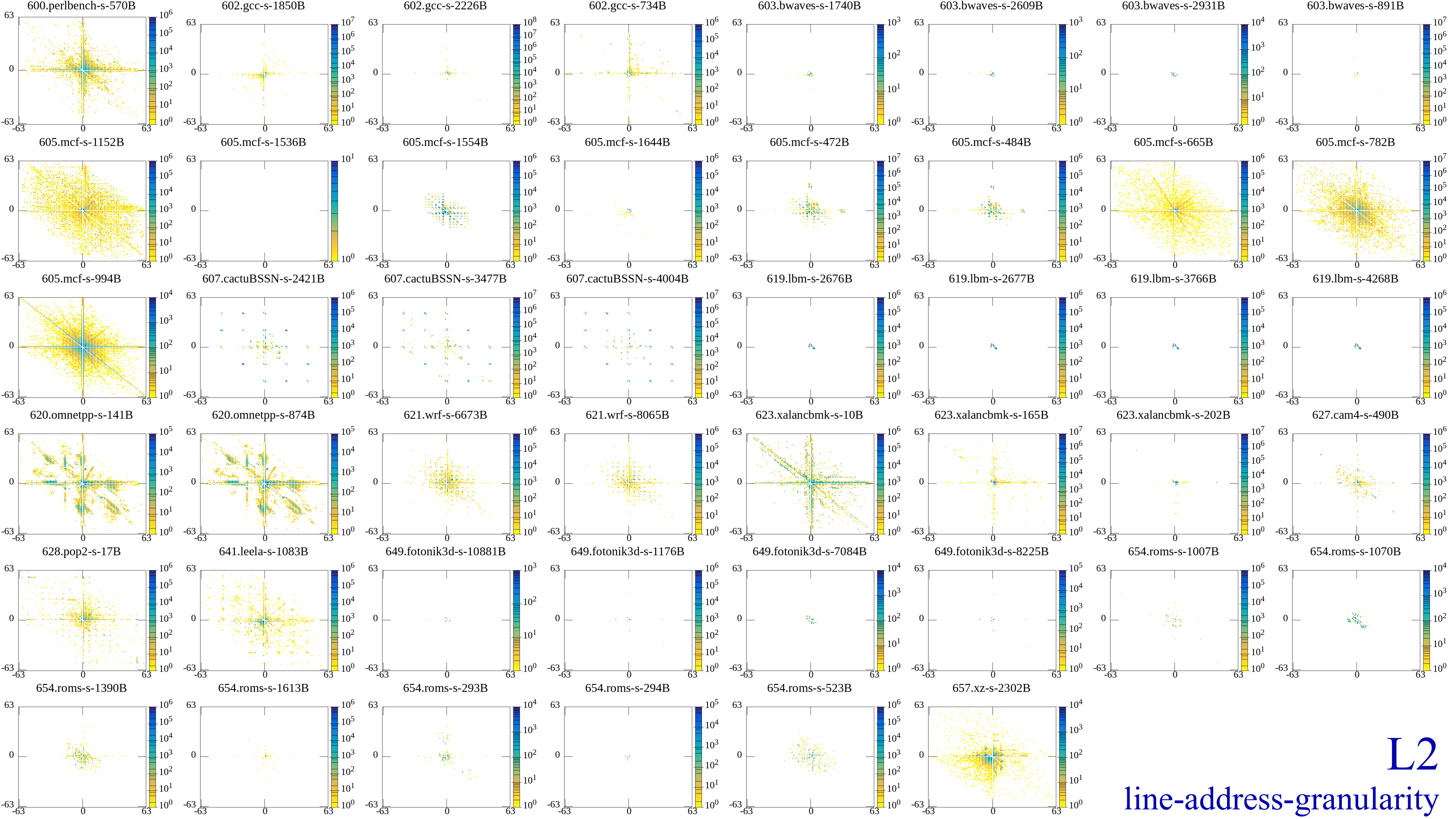
Thank you for your attention!

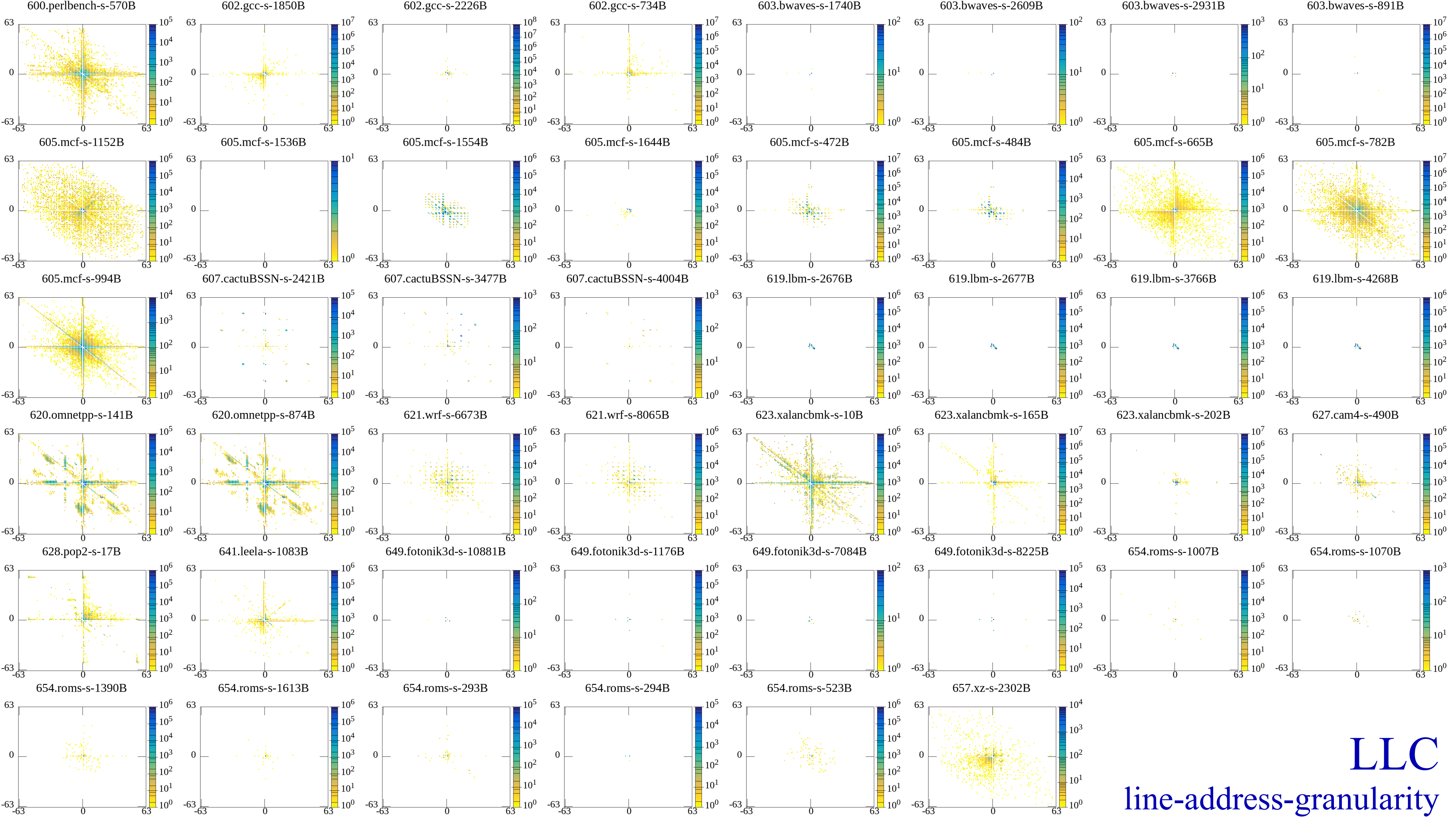
Questions?

Backup slides



L1
word-address-granularity


600.perlbench-s-570B
602.gcc-s-1850B
602.gcc-s-2226B
602.gcc-s-734B
603.bwaves-s-1740B
603.bwaves-s-2609B
603.bwaves-s-2931B
603.bwaves-s-891B
605.mcf-s-1152B
605.mcf-s-1536B
605.mcf-s-1554B
605.mcf-s-1644B
605.mcf-s-472B
605.mcf-s-484B
605.mcf-s-665B
605.mcf-s-782B
605.mcf-s-994B
607.cactuBSSN-s-2421B
607.cactuBSSN-s-3477B
607.cactuBSSN-s-4004B
619.lbm-s-2676B
619.lbm-s-2677B
619.lbm-s-3766B
619.lbm-s-4268B
620.omnetpp-s-141B
620.omnetpp-s-874B
621.wrf-s-6673B
621.wrf-s-8065B
623.xalancbmk-s-10B
623.xalancbmk-s-165B
623.xalancbmk-s-202B
627.cam4-s-490B
628.pop2-s-17B
641.leela-s-1083B
649.fotonik3d-s-10881B
649.fotonik3d-s-1176B
649.fotonik3d-s-7084B
649.fotonik3d-s-8225B
654.roms-s-1007B
654.roms-s-1070B
654.roms-s-1390B
654.roms-s-1613B
654.roms-s-293B
654.roms-s-294B
654.roms-s-523B
657.xz-s-2302B
L2
line-address-granularity


600.perlbench-s-570B
602.gcc-s-1850B
602.gcc-s-2226B
602.gcc-s-734B
603.bwaves-s-1740B
603.bwaves-s-2609B
603.bwaves-s-2931B
603.bwaves-s-891B
605.mcf-s-1152B
605.mcf-s-1536B
605.mcf-s-1554B
605.mcf-s-1644B
605.mcf-s-472B
605.mcf-s-484B
605.mcf-s-665B
605.mcf-s-782B
605.mcf-s-994B
607.cactuBSSN-s-2421B
607.cactuBSSN-s-3477B
607.cactuBSSN-s-4004B
619.lbm-s-2676B
619.lbm-s-2677B
619.lbm-s-3766B
619.lbm-s-4268B
620.omnetpp-s-141B
620.omnetpp-s-874B
621.wrf-s-6673B
621.wrf-s-8065B
623.xalancbmk-s-10B
623.xalancbmk-s-165B
623.xalancbmk-s-202B
627.cam4-s-490B
628.pop2-s-17B
641.leela-s-1083B
649.fotonik3d-s-10881B
649.fotonik3d-s-1176B
649.fotonik3d-s-7084B
649.fotonik3d-s-8225B
654.roms-s-1007B
654.roms-s-1070B
654.roms-s-1390B
654.roms-s-1613B
654.roms-s-293B
654.roms-s-294B
654.roms-s-523B
657.xz-s-2302B
LLC
line-address-granularity

Markov chains from other benchmark traces