Lecture: Sequence Alignment Accelerators

- Topics: GenAx, GenCache, Darwin accelerators

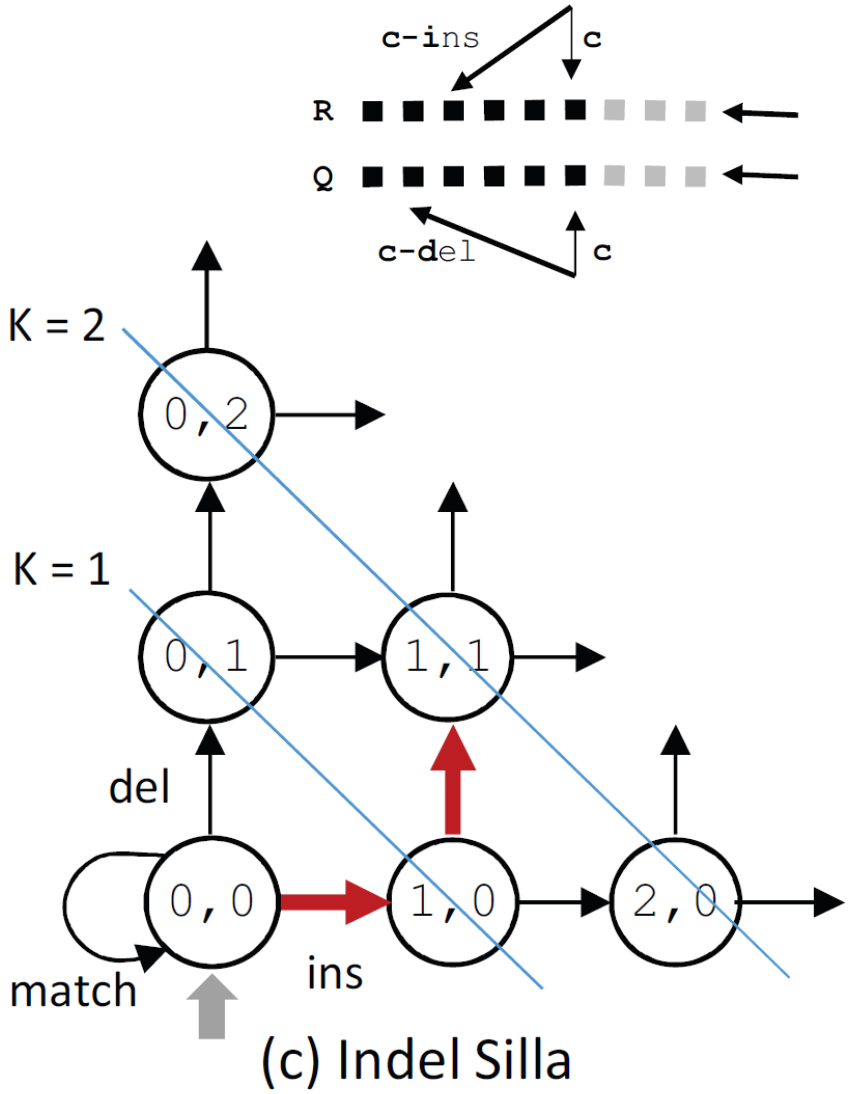
Sequence Alignment Basics

- Terminology: read, seed, reference
- Three main stages: seed selection, filtering, scoring
- 2nd gen sequencing: 100 bp (2% errors), 270 CPU hours
- 3rd gen sequencing: 10K bp (15% errors), 1300 CPU hours (PacBio 15% errors, 50% coverage, 99.99% accuracy)
- 2nd gen algorithms: SNAP, BWA-Mem

GenAx

- Accelerator for 2nd gen sequencing
- New finite state machine to handle scoring
- New caching/CAM data structures for filtering (part of the SMEM seeding algorithm)

Scoring with Insertions and Deletions (2D Silla)



(c) Indel Silla

clock		
0	ins	i = 0, d = 0
1	match	i = 1, d = 0
2	del	i = 1, d = 0
3	match	i = 1, d = 1

Alignment:

- A x B C D
y A - B C D

Details

- An FSM with hardware for each state; state defined by (ins,del)
- When input chars are received, we move to one of the next states; accordingly pointers to the input strings are adjusted
- With K errors, need $O(K^2)$ states
- Book-keeping to track the indels
- In example, cycle c refers to the input chars coming in
- Note how data is fed; broadcast of read/ref along rows/cols, the comparison at a node is also fed along the North-East diagonal

3D Silla

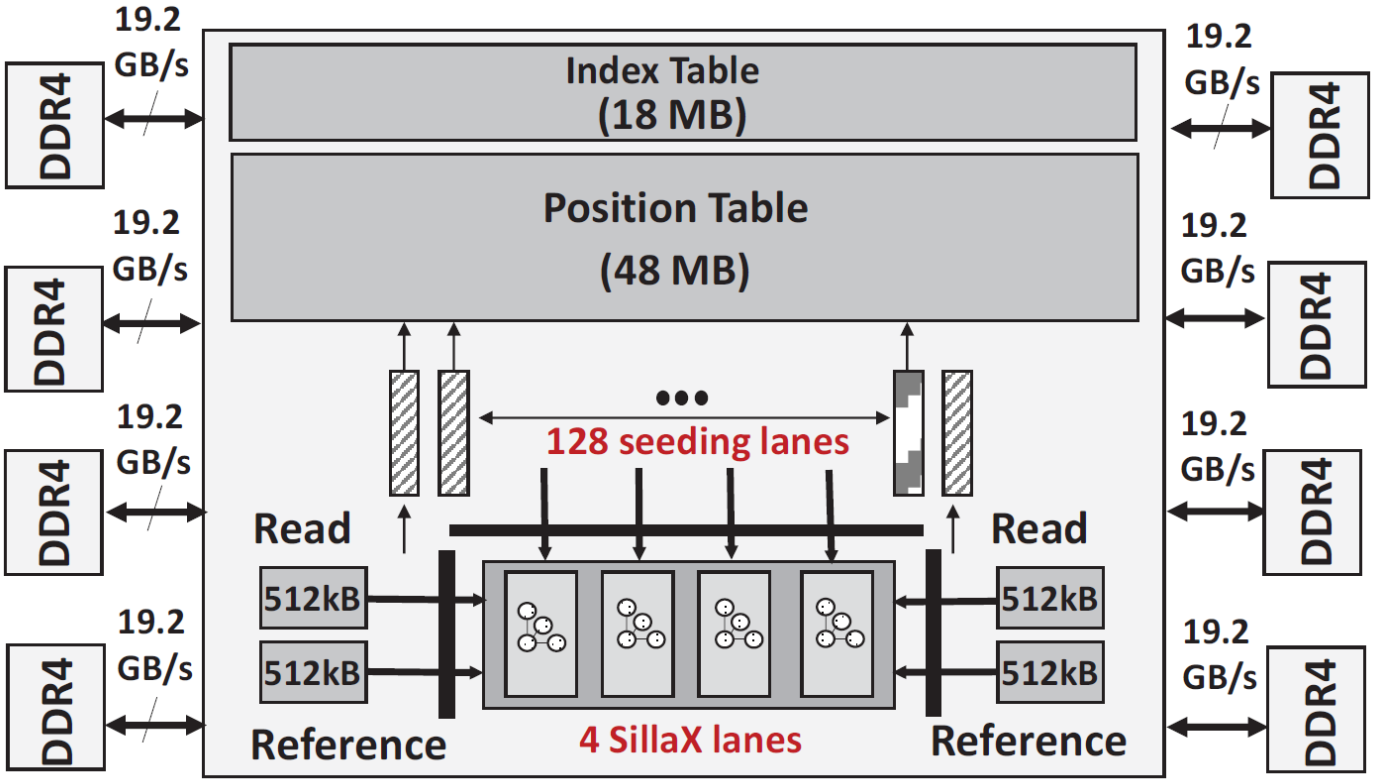
- Can introduce a third dimension for substitutions
- Harder to map to hardware; therefore, the 3rd dimension is collapsed into the 2D Silla
- State `i,d|s` has same behavior as state `i+1,d+1|s-2`

SMEM Algorithm for Seeding/Filtering

- Take a seed of size k in the read; look up the hash table to obtain a list of positions P for that seed in the reference
- Take the next seed of size k; look up the hash table to get a list of positions P'; intersect P and P' to get the best fits; keep doing until the whole read has been analyzed
- If the intersection is null, keep shortening the seed until you have a non-null intersection – that is the maximal matching seed
- Most of the operations involve hash table look-ups and set intersections; they introduce 512-wide CAMs (since most hash table look-ups return < 512 locations)

GenAx Architecture

- Tiling so that a 66MB slice of hash table fits in SRAM buffer
- 4 SillaX lanes that score the locations coming out of SMEM
- SMEM computed on 128 seeding lanes, that are essentially CAMs

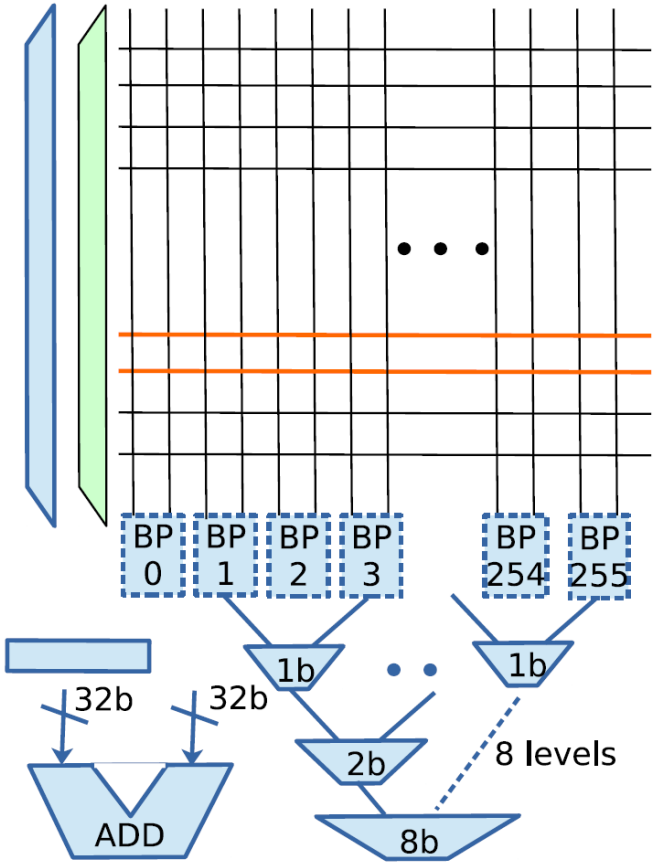


GenCache

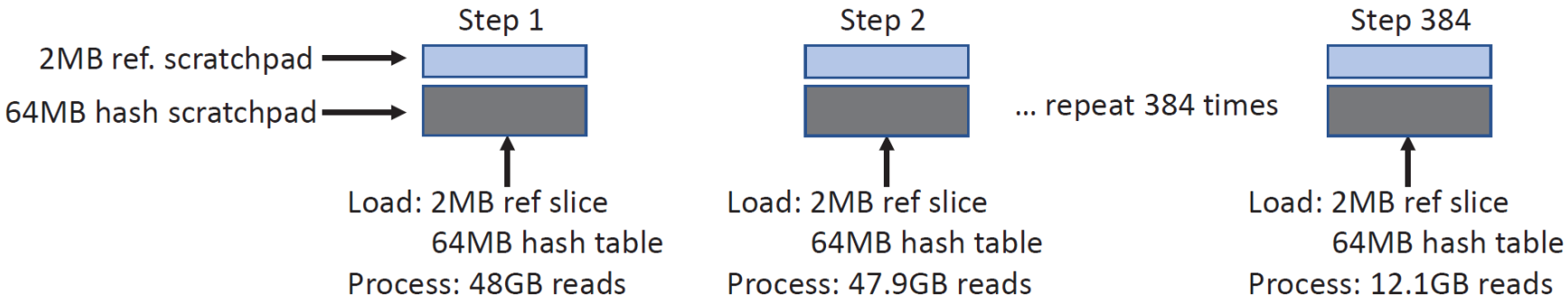
- Uses in-cache operators for high-parallelism bitline checks
- Modifies the algorithm so we're doing more filtering with less work (?) and sending fewer locations to the time-intensive scoring algorithm (unmodified)

In-Cache Operators for Filtration

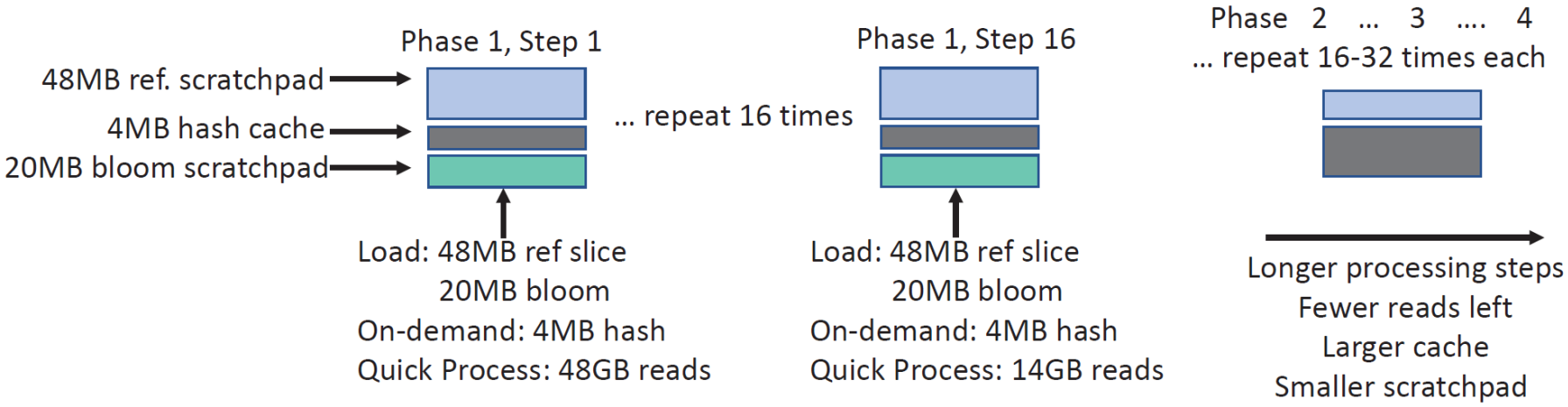
- New bitline operators that deal with 2 bits at a time
- Useful for various parallel filtering operations
- Hamming distance = 0 indicates perfect match
- SHD with 1 or 2 shifts useful for near-perfect matches
- Need shifts, gates, adders
- Next, must re-org the algorithm to exploit operators



GenAx vs. GenCache Computation Pipeline



(a) GenAx computation pipeline.



(b) GenCache computation pipeline.

GenCache Principles

- 80% of reads have perfect matches; 15% have 1-2 error matches
- Handle perfect matches in Phase 1, 1-error matches in Phase 2, 2-5 error matches in Phase 3, rest in Phase 4 similar to GenAx
- More space for the reference so more in-cache parallelism
- Reduces redundant work to scoring; reduces read fetches, same reference fetches
- Main drawback is that hash table doesn't fit
- About 90% of seeds have 0 or 1 entry in the hash table; track these seeds in a Bloom Filter; when looking for perfect and near-perfect matches, really helps to find a seed that hits in the Bloom Filter
- Need hw/sw co-design for accelerated genomics

Darwin

- New algorithms and architecture for 3rd gen sequencing
- New high-error-aware filtering algorithm
- Tiled scoring to reduce hardware overheads (heuristic to reduce overhead because errors k are high, which in turn makes banded scoring or GenAx O(k^2))

Filtering Algorithm D-SOFT

- Split the reference into 32M 128-wide bins
- Partition the read into several short seeds; each hash table look-up gives us several locations; mark red dots
- A winning location is one with the most matching bases
- Memory bound; 64MB counters in buffers and hashes in memory

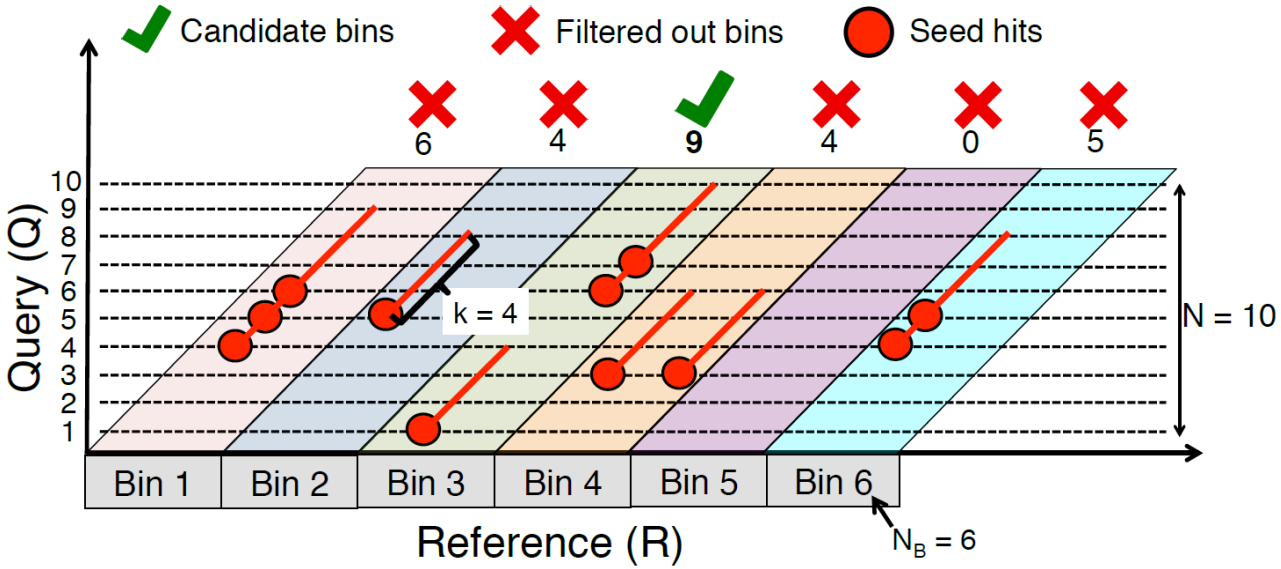
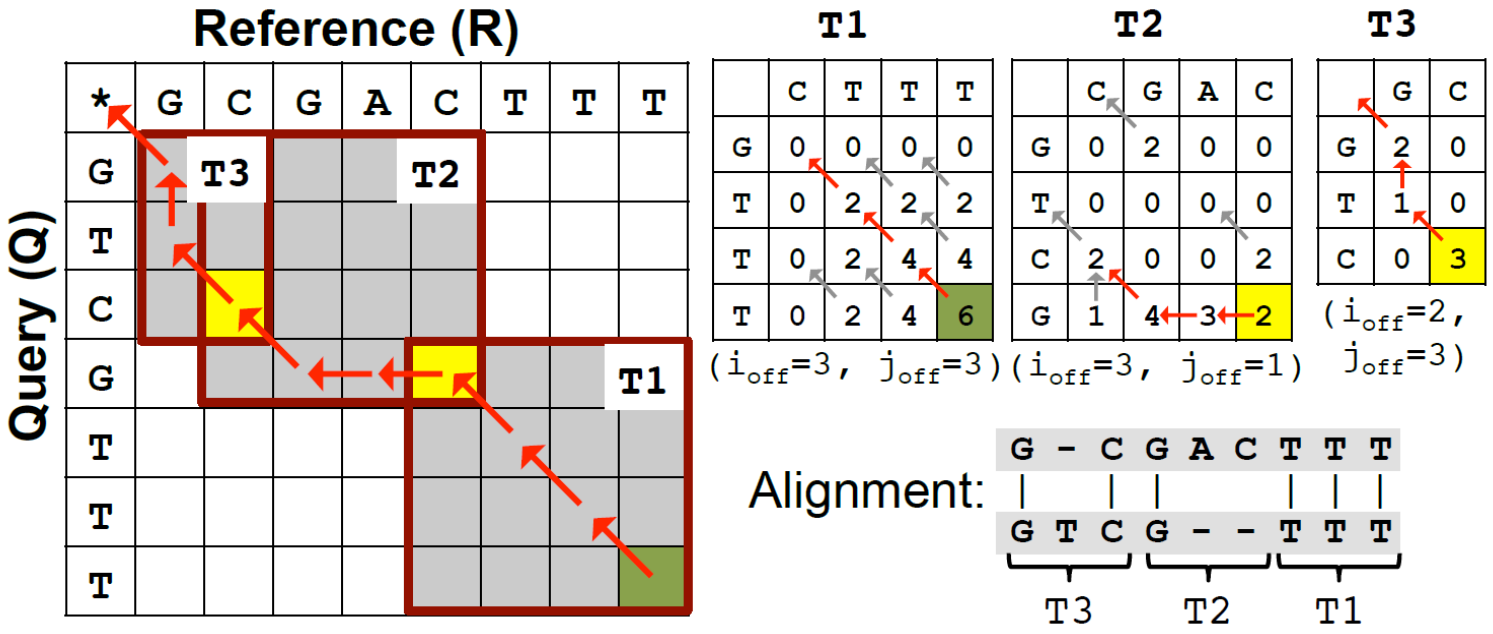


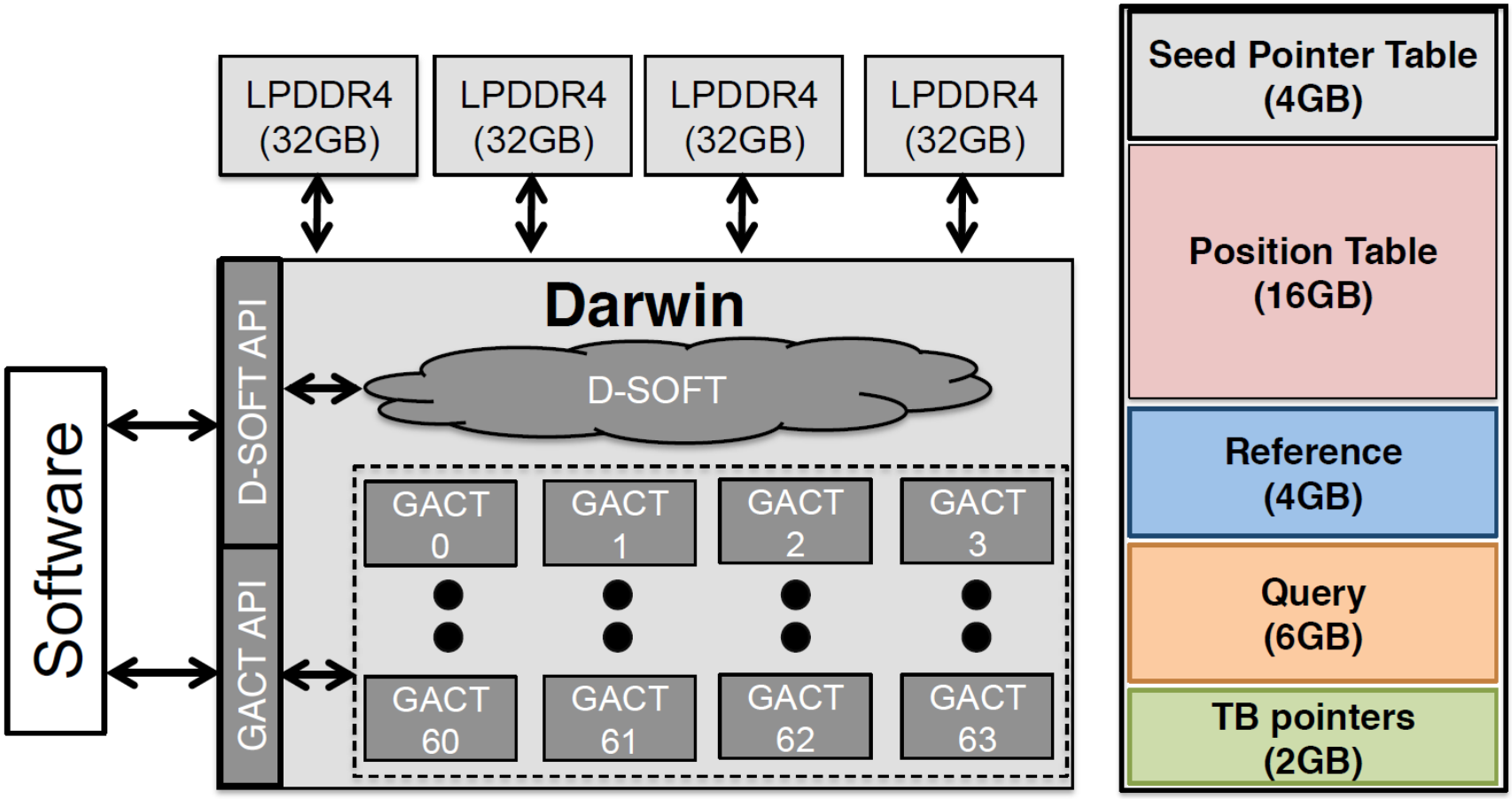
Figure 2: Illustration of D-SOFT algorithm for k=4, N=10, h=8, N_B=6.

Scoring Algorithm GACT

- Doing one tile of scoring computation at a time; then moving on to a next overlapping tile that includes the best score so far; tile size T and overlap O are selected such that good scores aren't missed; implemented with systolic array of size T^2.



Darwin Overview



References

- “GenAx: A Genome Sequencing Accelerator”, D. Fujiki et al., ISCA 2018
- “GenCache: Leveraging In-Cache Operators for Efficient Sequence Alignment”, A. Nag et al., MICRO 2019
- “Darwin: A Genomics Co-Processor Provides up to 15,000x Acceleration on Long Read Assembly”, Y. Turakhia et al., ASPLOS 2018