Exposing **memory level parallelism**

at **compile time**

by **slicing** indirections `C[B[A[i]]]`

and **decoupling** memory accesses.

DPREF: Decoupling using Dead Code Elimination for Prefetching Irregular Memory Accesses

Konstantinos Sotiropoulos, Jonas Skeppstedt, Angelos Arelakis, Per Stenström
Chalmers University of Technology, Lund University, ZeroPoint Technologies

DCE for slicing dead code

Dead Code Elimination (DCE) is a compiler optimization used to remove **ineffectual** code. "Pre-live" statements like line 6, which affect system state, form the slicing basis for DCE.

```
1  void func(int *A, int B) {
2    int C = B % 5; // data dependency of line 6
3    if (C == 0)    // control dependency of line 6
4      B = 0;
5    else
6      *A = C;      // "pre-live"
7  }
```

Slicing the Sparse Matrix-Vector multiplication

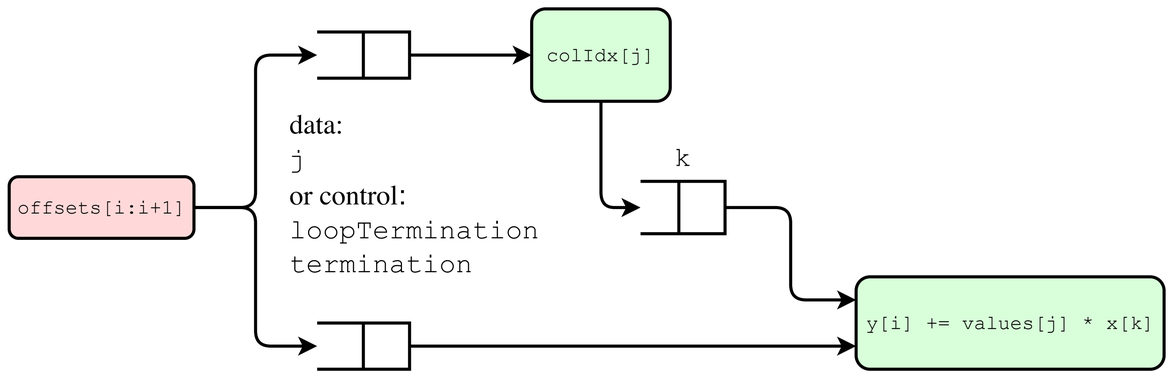
DCE can evolve into a **flexible slicing technique** by redefining what is considered as "pre-live". The `pragma` in line 9 characterizes line 10 as "pre-live", with the dependency on `j` considered satisfied.

```
1  // Compute y = Ax where A is a sparse matrix
2  // represented by: offsets, colIdx and values
3  void SpMV(...) {
4    // for every row of the matrix
5    for (int i = 0; i < N; i++) {
6      y[i] = 0;
7      // for every non-zero column of row i
8      for (int j = offsets[i]; j < offsets[i+1]; j++)
9        #pragma dpref sliceOn(j)
10       y[i] += values[j] * x[colIdx[j]];
11   }
12 }
```

Shaded in green is exclusively "live" code, shaded in red is exclusively "dead" code, and code in yellow is common to both.

Slice arrangement

Slices, mapped to threads, are arranged in a dataflow manner using queues for inter-slice communication, which carry both data and control tokens.



Microarchitectural challenges

The slicing requires ISA-visible queues, which should support:

- low-latency, fine-grained and blocking push/pop operations
- low-latency checks for control flow tokens

Related Work

Manocha, A., Sorensen, T., Tureci, E., Matthews, O., Aragón, J.L., and Martonosi, M. 2021. **GraphAttack: Optimizing Data Supply for Graph Applications on In-Order Multicore Architectures** ACM Transactions on Architecture and Code Optimization (TACO) 18, 1–26.

Nguyen, Q.M. and Sanchez, D. 2023. **Phloem: Automatic Acceleration of Irregular Applications with Fine-Grain Pipeline Parallelism** 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA).

CHALMERS UNIVERSITY OF TECHNOLOGY

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM