Workshop on MLIR for HPC, 21 October 2019

Compiler-managed Cache on Commodity Hardware

Chen Ding
Professor
University of Rochester
Rochester, New York, USA

Joint work with Dong Chen

Memory Problems

- The perennial bottleneck
  - Memory Wall [Wulf and McKee, 1995]
  - Memory Bandwidth Bottleneck [My dissertation, advisor Ken Kennedy, 1999]
  - Power Wall ~2005
  - Locality Wall [Kogge, MEMSYS 2017 keynote]
    - Data movement problems take center stage
- Memory complexity
  - Organization: size, topology, sharing
    - To support massive parallelism
  - Objectives: latency, throughput, power, energy, lifetime
- Compartmental and heuristic solutions are increasingly brittle
  - Need new abstractions to reliably tackle complexity

Lease Cache [ASPLOS'19]

- Each access is accompanied by a lease
  - Measured by the number of accesses
- The cache stores the data block for the duration of the lease
  - Let the currently accessed data block be B
  - If B's next access happens before the lease expires
    - It is a cache hit, and the lease is renewed
  - Otherwise, B is evicted at the end of the lease
- Two analogies
  - Lease cache is allocated as a heap is allocated
    - A lease is a lifetime in cache
  - Automatic water faucet at an airport restroom
- Lease cache is collaborative [Wang, McKinley, Rosenburg, Weems, PACT'02]

Statistical Caching Using the Lease Cache [ASPLOS'19]

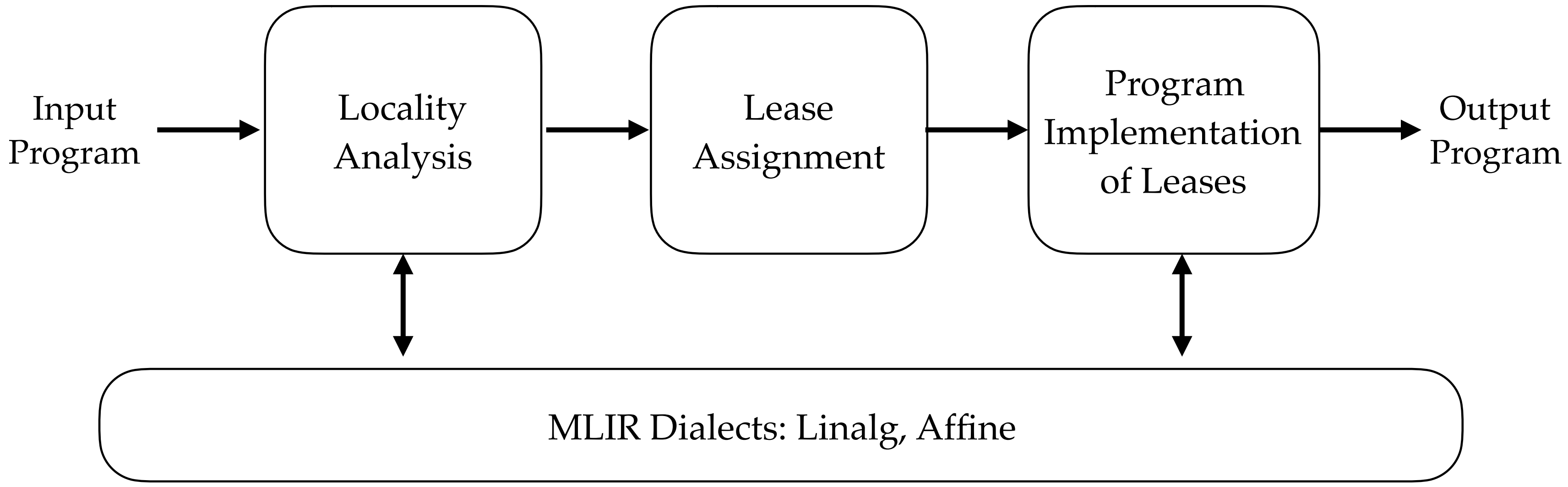
- Locality represented by the distribution of forward reuse intervals
  - Future prediction based on collective behavior
    - Instead of individual accesses
    - Statistical prediction subsumes precise prediction
- Optimal statistical caching
- Degenerate case: uniform lease for all accesses
  - It approximates the performance of LRU
- Other benefits
  - Guaranteed worst case access time in real-time systems
  - Defense against side-channel attacks
- One problem: the lease cache may not be fixed size
- This talk: program-level leases on x86 processors

An Example Program with Leases [LCPC'19]

```
# Leases:
# 1 for ref1 accesses
# 2 for ref2 accesses
# 3M for (C-1)/M of ref3 accesses
# 0 for the rest of ref3 accesses
REPEAT FOREVER
   DO I = 1, M
      A = A + B(I) # ref1-ref3
   ENDDO
END REPEAT
```

- Assume unit size cache block, cache size $C < M$ the array size
- Both A, B are cached

Compiler Assigned Reference Leases (CARL)


Input Program
Locality Analysis
Lease Assignment
Program Implementation of Leases
Output Program
MLIR Dialects: Linalg, Affine

Locality Analysis

- A relational theory of locality [TACO'19]
  - Access locality, e.g. reuse distance
  - Timescale locality, e.g. the footprint [ASPLOS'13]
  - Cache locality e.g. the miss ratio
- Timescale locality
  - Working-set size of all window lengths
  - Computed from the reuse intervals
    - Denning recursion [CACM'72]
      - Explanation for finite-length traces [LCPC, 2018 in University of Utah]
    - Xiang formula [PACT'13]
    - Formal relations [TACO'19]
- Can it be computed using MLIR dialects?

Program Implementation of Leases

- Software control of cache eviction
  - cflush and cflushopt on x86
  - Designed to support persistency
  - Used by HP's Atlas system [Chakrabarti, Boehm, Bhandari OOPSLA'14]
- Lease implementation
  - Static eviction insertion
    - Related problem: software prefetching [Callahan, Kennedy, Porterfield ASPLOS'91; Mowry, Lam, Gupta, ASPLOS'92]
      - Ahead of time prefetch insertion (by k iterations)
    - Preliminary transformation: scalar replacement [Carr, Kennedy TOPLAS'94]
    - Can MLIR dialects help lease implementation?
  - Dynamic eviction for irregular code

Summary

- Compiler leasing of cache memory
  - Directly leveraging program knowledge of precise and statistical future
- Compiler implementation
  - Program analysis to identify forward reuse intervals
  - Optimal lease assignment algorithm
  - Eviction insertion
    - Using cflush and cflushopt on x86
- Research questions
  - How is MLIR as the basis?
  - Accuracy of the analysis and efficiency of eviction control
  - Variable size cache demand
  - Direct hardware support on accelerators (tomorrow's talk)