Autotuning matrix transposition

Lai Wei
John Mellor-Crummey
Differences among architectures growing

- Optimized code varies for different architectures

Bandwidth achieved by different configurations of matrix transposition on

- **Davinci** (Intel Xeon X5660)
- **Biou** (IBM Power7)
Machine model as a solution

- Use machine models to guide efficient code generation/optimization
  - Hardware parameters
    - Number of threads
    - Cache size, cache line size
  - Empirical data
    - Run time
    - Hardware performance counters
A naive 2D matrix transpose implementation

for (i=0; i<LENGTH; i++)
    for (j=0; j<WIDTH; j++)
        dst[j][i] = src[i][j]
A naive 2D matrix transpose implementation

Memory

Cache

Holds 4 cache lines

Each line holds 4 elements

Each write is a cache miss
Tiling to enhance spatial reuse

Memory

Cache

Holds 4 cache lines
Each line holds 4 elements

Reduces the number of cache misses
Getting the optimized code

- Tiling
  - The best tile sizes?
  - Multi-level tiling?

- Other optimizations
  - SIMD? Non-temporal store?
  - In-cache buffer? Buffer sizes?
  - Software prefetching? Prefetch depth?

- Needs a good choreography among these optimization settings
Framework for matrix transposition

- A framework that generates optimized matrix transposition code for various platforms.

  - Code generation rules
    - generate code with different optimization configurations

  - Auto-tuner
    - choose among implementation alternatives
Code generation

- User input
  
  array element type = float
  Function = {
    function name = matrixTransposition2D
    number of dimensions = 2
  }

- Auto-tuner input
  - Optimization parameters

- Output
  - C functions that performs matrix transposition
Auto-tuner

- Use **machine parameters** to prune the search space

- Direct code generation rules to generate code with different parameters, collect empirical data

- Pick the best optimization configuration
Experimental result

- We compared the bandwidth achieved by our matrix transposition code with
  - Array copy
    - STREAM benchmark (practical upper bound)
    - Memcpy
    - Strided copy
  - Matrix transposition in TCE
Single-thread performance

On Davinci (Intel Xeon X 5660), with N x N double matrix
On Biou (IBM Power7), with 32 threads running on N x N float matrix.
Influence of optimizations

On Davinci, with 38400 x 38400 float matrix
On Davinci, with 6 threads running on 38400 x 38400 float matrix
On Biou, with 32 threads running on 38400 x 38400 float matrix
Conclusions & future work

- Optimized code varies for architectures, machine model as a solution
- Our framework generates efficient matrix transposition code across platforms

- Achieving memory bandwidth closer to STREAM
- Generate efficient library code useful across all matrix sizes
- Improve autotuner efficiency & configurability