With DAWS: Divergent Code within 4% of Locality Managed Code with no Programmer Input

Divergence-Aware Warp Scheduling

Programmability

- Simple, But Divergent
- Less Divergence, But More Complicated

Hardware aware of code locality can take advantage of it without needing the programmer

Example Operation

Cache is 4 entries, 1288 lines and fully associative.
By Time₀, warp 0 has entered loop and loaded 4 lines into cache. By Time₁, warp 0 has captured spatial locality, DAWS measures footprint. Warp 1 is prevented from scheduling as DAWS predicts it will oversubscribe cache. By Time₂, warp 0 has accessed 4 lines for 32 iterations and loaded 1 new line. 3 lanes have exited loop, decreasing footprint. Warp 1 and warp 0 are allowed to capture spatial locality together.

Results

- Greedy-than-oldest (GTO)
- Cache-Conscious Wavefront Scheduling (CCWS)
- Profile based Static Wavefront Limiting (Best-SWL)
- Divergence-Aware Warp Scheduling (DAWS)

Performance vs. Other Schedulers on Cache-Sensitive Applications

26% Speedup over Cache-Conscious Wavefront Scheduling

Example Compressed Sparse Row Kernel

```
void sum_rowcsr(float* A, ...) {
    float sum = 0;
    int i = C[tid];
    while(i < C(i+1)) { <- Divergent Branch
        sum += A[i]; <- Divergent (or Uncoalesced) Load
        ++i;
    }
}
```

Programmability Case Study Results

Diverged Code vs. Locality Managed Code

DAWS: Divergent Code is within 4% of Locality Managed Code with no Programmer Input

Proactive Predictions: Reduce Cache Misses

Branch Divergence Awareness: Increases Multithreading when Appropriate

Timothy G. Rogers¹, Mike O’Connor², Tor M. Aamodt¹
¹The University of British Columbia ²NVIDIA Research