Divergent loads a major problem when programming GPUs

- Common in irregular applications
- Programmer encouraged to restructure program
- Our work asks: What if they didn’t have to?

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 Time2, warp 0 has captured spatial locality, DAWNS measures footprint. Warp 1 is prevented from scheduling as DAWNS 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.

Programmability

Simple, But Divergent

Less Divergence, But More Complicated

Divergent Code vs. Locality Managed Code

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

Results

Performance vs. Other Schedulers on Cache-Sensitive Applications

Proactive Predictions: Reduce Cache Misses

Branch Divergence Awareness: Increases Multithreading when Appropriate

26% Speedup over Cache-Conscious Wavefront Scheduling

Example Compressed Sparse Row Kernel

```c
int C[]={0,64,96,128,160,160,192,224,256};
void sum_row_csr(float* A, ...) {
    float sum = 0;
    int i = c[tid];
    while(i < C[tid+1]) {
        Divergent Branch
        sum += A[i];
        Divergent (or Uncoalesced) Load
        i += C[tid+1];
    }
}
```