Sector Cache Optimizations for the K Computer
Swann Perarnau, Mitsuhisa Sato
RIKEN AICS, University of Tsukuba

Sector Cache on the SPARCVIIIIfx

- On-demand split of the shared L2 cache.
- User-controlled mapping between accesses and sectors.
  - Isolation of thrashing accesses.
  - Select and keep useful data in cache.

Our Goals

- Assess applicability of the sector cache to optimize HPC applications.
- Study potential optimization strategies.
- Help users find good sector cache optimizations.
- Aim for as much automation as possible.

Issues

- Low-level compile-time API.
- Hard to predict impact on performance.
- Little support from compiler and performance analysis tools.

Current Sector Cache API

double myarray[NSIZE];
double otherarray[NSIZE];

void mywork(void)
{
    int i;
double sum = 0;
#pragma statement cache_sector_size 1 11
#pragma statement cache_subsector_assign myarray
for(i = 2; i < NSIZE-2; i++)
{
    // myarray in sector 1
    sum += myarray[i-2] + myarray[i-1] +
           myarray[i] + myarray[i+1] +
           myarray[i+2] + otherarray[i];
}
}

Locality Measurements

Reuse Distance: for a memory access, the number of unique memory locations touched since the previous access to the same location.
- An architecture-independent measure closely related to the cache misses triggered by an application.
- Use it to measure consequences of isolating one structure by itself with the sector cache.

Validating the Framework: a toy example

Reuse Distances of Each Structures

Optimizing the NAS Parallel Benchmarks

Benchmark | Function | Isolated Variables | Sector Size | Miss Reduction (%) | Runtime Reduction (%) |
-----------|----------|--------------------|-------------|---------------------|-----------------------|
LU         |           | a,b,c,d            | (2,10)      | 66                  | 3                     |
           |           | d                   | (2,10)      | 76                  | 10                    |

Acknowledgements

Part of the results were obtained by early access to the K computer at the RIKEN AICS. This work was supported by the JSPS Grant-in-Aid for JSPS Fellows Number 2402711.