SAND2015-6811D

Vector Seeker Block Extensions

Sandia National Laboratories
G. Carl Evans, Simon Hammond (Advisor), David Padua (Advisor)
Sandia National Laboratories, New Mexico and University of Illinois, Urbana Champaign

Vector Seeker

Vector Seeker is a tool to help compiler designers and programmers to deal with vectorization.

- What vectorization opportunities are missed by the compiler ?
- Where is there potential opportunities for manual transformations ?
- What opportunity is there for vector parallelism, or how much headroom is there?

Tracing

At the core Vector Seeker uses tracing to find the depth in the dynamic dependence graph that each instruction is executed at and declares two instances of the same static instruction to be vectorizable if they are are the same depth.

Vector Memory

Vector memory is how the dependance on loop variables is broken. Consider the simple loop to the right. No instruction happens at the same time due to the true dependence on the loop variable i. To break this we partition the graph into two parts one coming from vector memory and the other from non-vector memory. The non-vector memory can be completely removed. This gives the pruned graph.

Implementation

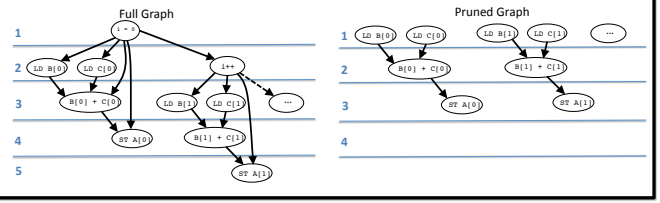
Vector Seeker is build on top of PIN and makes a function call for each instruction in the original program. This combined with the locality destruction of the direct has table shadow memory severely limit the performance of the tool.

Greedy Instrument

This is computed using the Greedy Instrument algorithm. It is called for each instruction in the input program and updates the two key global structures. First Shadow Memory(SM) that contains the depth which the value in the same main memory address was computed at or ⊥ if the value comes only from non-vector memory. The second structure the Result Vector(RV) is a map from instructions and depths to the number of times that instruction was encountered a that depth.

```
D← max(all source operand shadow memory)
I ← instruction address
if I is a vector allocation then
        for all addresses A in allocation do SM[A] = D + 1
else if I is a vector deallocation then
        for all addresses A in vector deallocation do SM[A] = ⊥
else if instruction = simple load or store with address A then
        SM[A] ← D
else if D != ⊥ then
        SM[destination address] ← D + 1
        RV[I][D +1] ← RV[I][D +1]+1
else
        SM[destination address] ← ⊥ or 1 if vector location
```

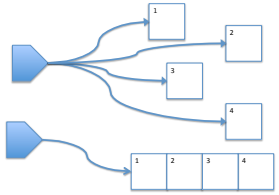
```
for(i = 0; i < N; i++)
  A[i] = B[i] + C[i];
```



Block Extensions

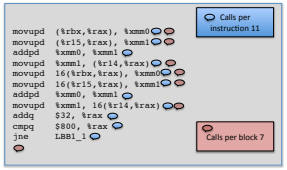
Shadow Memory Blocking

- To improve memory usage and localty Shadow Memory was changed from a direct map to a blocked map. This allows Vector Seeker to take advantage of the locality in the original code that is instrumented.
- Significant overhead in pointers is also saved by requiring only one index per cache line
- Many locations only ever contain values of slight depth. To take advantage of this we also start with only allocating a byte per location and reallocate when larger memory cells are required to hold the depth



PIN Basic Block Blocking

- To reduce the number of calls Vector Seeker was changed to use PIN basic blocks. This way several instructions can be executed in a single instrumentation call.
- To avoid doing computations in the original program the addresses accessed are caputred and then reused when the system executes the basic block.

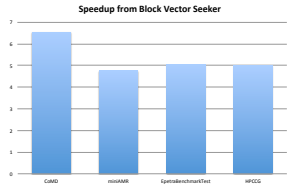


Experiments

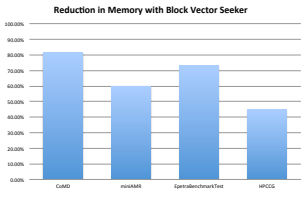
Experiments were run on a subset of the Mantevo Suite of proxy apps.

- CloverLeaf 1.1(Block Version only)
- CoMD 1.1
- miniAMR 1.0
- Epetra Benchmark Test 1.0
- HPCCG 1.0

Small inputs were used for these apps and we present here the performance comparison in memory and time of the improved Vector Seeker with the original Vector Seeker.
Finally the summary vector results on the blocks is presented for each of these applications.



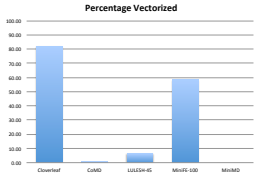
Application	Static Blocks			Dynamic Blocks	
	Total	Vector	Non-Vector	Vector	Total
CloverLeaf	12545	648(8.42%)	7694	290369(9.40%)	3089025
CoMD	7481	101(2.22%)	4554	28261693(3.22%)	877158775
miniAMR	4055	39(2.26%)	1724	9624933(3.79%)	254131398
Epetra Benchmark	8900	6(0.17%)	3589	1118861(1.29%)	86887414
HPCCG	7322	12(.34%)	3488	78645772(5.87%)	1339131230



Vectorizability

Compiler Auto Vectorization

- Work measuring the vectorizablity of a subset of the Mantevo mini-apps showed that autovectoriation by the compiler has mixed results.
- In this case using the Intel Compiler Suite 15.2 we see that two applications have significant success at vectorization but the others receive minimal benefit.
- These tests showed a remaining issue in Vector Seeker, while the performance improvements has helped the current method of selecting vector memory is too course. Future work will address this limitation.



Center for Exascale Simulation of Plasma-Coupled Combustion

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.