Performance Characteristics of Cache Oblivious Implementation Strategies for Hyperbolic Equations on Opteron Based Super Computers

May 11, 2005

David Hensinger,
Chris Luchini,
Sandia National Laboratories
Volker Strumpen,
Matteo Frigo,
IBM Austin Research Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Cache Oblivious Strategies for Hyperbolic Problems

- Cache oblivious strategies attempt to minimize data motion in favor of floating point operations
- Calculations can be memory bound
  - Problem size is Gigabytes
  - Cache size is Megabytes
  - Register size is Kilobytes

Cache Oblivious Strategy

- Exploit topological data locality to re-use in cache data
- Recursively bisect solution domain in time and space
- Advance sub-domains multiple time steps during each sub-domain traversal
- Entire domain is traversed once while moving forward n steps

Cache Oblivious Requirements

- Advancement of problem depends on local data x (t+1) = f(x(t))
- A kernel optimized for floating point operations
- A method of managing multiple time states
  - Toggle arrays
  - Boundary passing

Multiple Time States Managment

- Toggle Arrays
  - Duplicates time dependent quantities
  - Increases storage requirement
  - Simple to implement
- Boundary Passing
  - Save boundary information
  - Propagate through space to neighboring domains
  - Complexity increases with spatial dimension

Toggle Arrays Example
SIGMA = 1, K=3

Building Up	Filling In
0000000	0113110
0000000	0022200
0111110	1113111
0000000	0022200
0111110	1113111
0022200	2222222
0113110	3333333
0022200	2222222

Sandia National Laboratories

Test Kernels

- 3D Arbitrary Work
  - Accumulate “P” from all neighbors
  - Accumulate “RT” from face neighbors
  - Calculate “T”
- 2D Lagrangian Hydrodynamics
  - Kernel version of ALEGRA 2D
  - Single material
  - Ideal gas
- 2DLax-Wendroff
  - Optimized
  - Advection solution

Optimizations

- Increase work per kernel with “V” (Voodoo) parameter
- Unroll Loops

Results

- **Arbitrary Work Kernel Results**

Table 1. Mflop/s for 3D Arbitrary Work Kernel, K= 8.

	Xeon	*Opteron*	*Power5*
Iterative/fits in L1	800	920	1188
Iterative	480	650	650
cache oblivious V=1	660	568	690
cache oblivious V=100	590	690	800
Wrong Way iterative	130	151	90
Wrong Way cache oblivious V=1	590	740	570
Wrong Way cache oblivious V=100	650	800	590

Results

- **Lagrangian Hydrodynamics Kernel Results**

Table 2. Mflop/s for 1,000,000 element 2D Lagrangian hydrodynamics problem with K=10.

	Xeon	*Opteron*	*Power5*
Iter/fits L1	538	810	541
Iterative 6.0×10^9 flops	463	605	494
CO V=1 7.3×10^9 flops	428	690	520
CO V=100 6.8×10^9 flops	446	739	539

Sandia National Laboratories

Results

- **Lax-Wendroff Kernel Results**

Table 3. Mflop/s for 2D Lax-Wendroff solutions with K=100.

	Xeon	*Opteron*	*Power5*
Unoptimized Iterative fits in L1 Cache	70	520	420
Optimized Iterative fits in L1 Cache	1050	1470	1774
Optimized Iterative	758	747	1280
Optimized Cache Oblivious	559	1410	1621

Conclusions

- Success requires a kernel optimized for floating point performance
- Cache oblivious traversal can mitigate effects of poor memory layout.
- Reducing calculations to a kernel was observed to produce performance improvements
- Performance results depend on the compiler/processor combination

1D Cache Oblivious Recursive Domain Decomposition

```
void walk(int t0, int t1, cut  cuts, float *** tsa){
  const int SIGMA = 1;
  int delt = t1-t0;
  if((delt == 1) || (((cuts.x1 - cuts.x0) ) < V)){
    basecase(t0,t1,cuts, tsa);
  }
  else if (delt > 1){
     if(2*(cuts.x1-cuts.x0)+(cuts.x1dot-cuts.x0dot)*delt  >=  4*SIGMA*delt){//dim0
cut
      cut save = cuts;
      int xm = (2*(save.x0+save.x1)+(2*SIGMA + save.x0dot + save.x1dot)*delt)/4;
      cuts = cut(save.x0,save.x0dot,xm,-SIGMA);walk(t0, t1, cuts, tsa);
      cuts = cut(xm,-SIGMA,save.x1,save.x1dot);walk(t0, t1, cuts, tsa);
      cuts = save;//restore configuration
    }
    else { //cut time
      int s = delt/2;
      cut new_cuts;
      walk(t0, t0+s, cuts, tsa);
                new_cuts  =  cut(cuts.x0  +  cuts.x0dot*s,  cuts.x0dot,  cuts.x1  +
cuts.x1dot*s, cuts.x1dot);
      walk(t0+s,t1,new_cuts,tsa);
    }
  }
}
```

Sandia National Laboratories