

Jacqueline Chame, Chun Chen, Mary Hall, Muhammad Murtaza. USC ISI
Paul Hovland, Jaewook Shin. Argonne National Laboratory

Goals

- Develop compiler-directed performance tuning technology targeting the Cray XT4 at Oak Ridge
- Achieve hand-tuned levels of performance on DOE Office of Science applications with compiler-generated code

Requirements

- New software technology to exploit performance of Cray XT4's multicore Opteron processors
- High single-node and single-core performance requires effective use of complex memory hierarchy and Opteron's SSE-3 instruction set
- High multi-core performance additionally requires parallelization and management of shared caches

Approach

- Combine compiler technology for model-guided empirical optimization for memory hierarchies and SIMD code generation

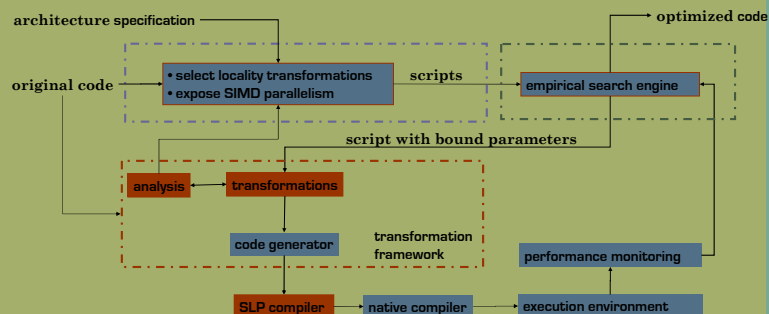
Compiler-Guided Empirical Optimization

- Generate alternative code variants (*scripts*) using code transformations
 - select profitable variants based on heuristics/models
 - target all levels of memory hierarchy
- Transformation framework (based on Omega)
 - imperfectly-nested loops, non-rectangular bounds
 - rich set of transformations including permutation, tiling, unroll-and-jam, fusion, distribution, copying
 - uniform representation facilitates composing transformations and search
- Empirical search
 - search among code variants
 - search parameter values of a code variant
 - reduce search space using constraints on parameter values and pruning heuristics

SIMD Parallelism on SSE-3

- Apply transformations to expose SIMD parallelism in innermost loops
 - SSE-3 registers as a level of memory hierarchy
- Shin's Superword-Level Parallelism (SLP) compiler
 - exploits exposed parallelism
 - manages alignment of loads/stores to 128-bit data types
 - manages limited SSE-3 register file (up to 16 128-bit registers)
 - superword replacement
 - register packing

TUNE framework: Model-guided Empirical Search and SIMD parallelism



DOE Office of Science Target Applications

- GTC
 - Massively parallel full torus gyrokinetic toroidal code.
 - Computationally intensive functions chargei and pushi show opportunities for SIMD parallelism
- S3D
 - Compressible Navier-Stokes solver coupled with an integrator for detailed chemistry, based on high-order finite differencing, high-order explicit time integration and conventional structured meshing
- MADNESS
 - Computational chemistry code that uses adaptive multiresolution methods in multiwavelet bases.
 - Performance of TUNE-generated code plus manually applied array contraction for single-precision use of SSE-3 of up to 1.23 speedup over hand-tuned code
- Nek5k
 - A spectral element code for simulating unsteady incompressible fluid flow with thermal and passive scalar transport
 - Core computation: Small rectangular dense matrix multiplications in three stages

Example code

(a) original code

```

new P[TK,TJ]
new Q[TK,TI]
DO K = 1, N
DO J = 1, N
DO I = 1, N
C[L,J] = C[L,J] + A[L,K] * B[K,I]
    
```

(b) transformation script

```

permute([L,J,K])
tile(0,J,TJ,JJ)
tile(0,I,TI,II)
tile(0,K,TK,KK)
datacopy(0,A,Q,II,transpose)
datacopy(0,B,P,JJ)
unroll(0,I,UI)
unroll(0,J,UJ)
    
```

(c) code variant with locality optimizations

```

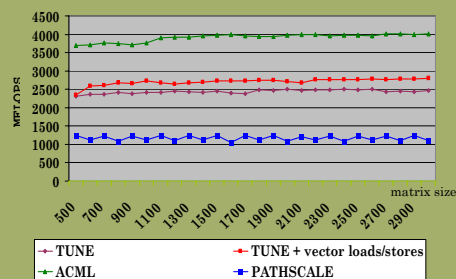
new P[TK,TJ]
new Q[TK,TI]
DO KK = 1,N,TK
DO II = 1,N,TI
Q[I:TK,1:TI] = A[II:II+TI-1,KK:KK+TK-1]
DO JJ = 1,N,TJ
P[I:TK,1:TJ] = B[KK:KK+TK-1,JJ:JJ+TJ-1]
DO I = II,min(II+TI-1,N)
DO J = JJ,min(JJ+TJ-1,N),4
T1 = C[L,J]
T2 = C[L,J+1]
T3 = C[L,J+2]
T4 = C[L,J+3]
DO K = KK,min(KK+TK-1,N)
T1 = T1+Q[K-KK+1,I-II+1]*P[K-KK+1,J-JJ+1]
T2 = T2+Q[K-KK+1,I-II+1]*P[K-KK+1,J-JJ+2]
T3 = T3+Q[K-KK+1,I-II+1]*P[K-KK+1,J-JJ+3]
T4 = T4+Q[K-KK+1,I-II+1]*P[K-KK+1,J-JJ+4]
C[L,J] = T1
C[L,J+1] = T2
C[L,J+2] = T3
C[L,J+3] = T4
    
```

(d) output of SLP compiler with reduction transformation

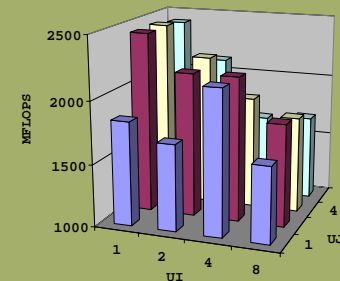
```

new P[TK,TJ]
new Q[TK,TI]
DO KK = 1,N,TK
DO II = 1,N,TI
Q[I:TK,1:TI] = A[II:II+TI-1,KK:KK+TK-1]
DO JJ = 1,N,TJ
P[I:TK,1:TJ] = B[KK:KK+TK-1,JJ:JJ+TJ-1]
DO I = II,min(II+TI-1,N)
DO J = JJ,min(JJ+TJ-1,N),4
T1[1:4] = {C[L,J], 0, 0, 0}
T2[1:4] = {C[L,J+1], 0, 0, 0}
T3[1:4] = {C[L,J+2], 0, 0, 0}
T4[1:4] = {C[L,J+3], 0, 0, 0}
DO K = KK,min(KK+TK-1,N),4
T1[1:4] = madd(T1[1:4],Q[K-KK+1:K-KK+4,I-II+1]*P[K-KK+1:K-KK+4,J-JJ+1])
T2[1:4] = madd(T2[1:4],Q[K-KK+1:K-KK+4,I-II+1]*P[K-KK+1:K-KK+4,J-JJ+2])
T3[1:4] = madd(T3[1:4],Q[K-KK+1:K-KK+4,I-II+1]*P[K-KK+1:K-KK+4,J-JJ+3])
T4[1:4] = madd(T4[1:4],Q[K-KK+1:K-KK+4,I-II+1]*P[K-KK+1:K-KK+4,J-JJ+4])
C[L,J] = sum(T1[1:4])
C[L,J+1] = sum(T2[1:4])
C[L,J+2] = sum(T3[1:4])
C[L,J+3] = sum(T4[1:4])
    
```

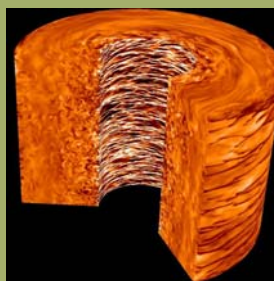
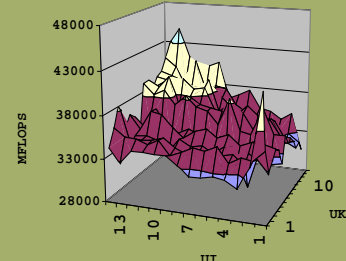
Matrix Multiply on Jacquard (NERSC)



Performance v. Unroll Factors for matrix size 1000, tile sizes TI=128, TJ=8, TK=512



Nek5k Performance v. Unroll Factors on Dual-Core AMD Opteron, 2.4 GHz, 1 MB L2 Cache, 64 KB L1 Data Cache



Turbulent flow field triggered by the magnetorotational instability. Sit.....
BGL platform at IBM Watson by Aleks Obabko (U. Chicago), Fausto Cattaneo (U. Chicago / ANL) and Paul Fischer (ANL).