Supercomputing for Everyone Series: Performance Tuning Summer School

X5: How to access data efficiently

Dr. Holger Brunst
Technische Universität Dresden, Germany

August 17–21, 2015
Goal

- tune memory access pattern
- study performance of nested loops
- measure floating point performance
Access to Blue Waters

- every participant receives a training account
- use your ssh client and login using traxxx account into bwbay.ncsa.illinois.edu
- create two ssh client sessions
  Unix: ssh -C -X <traxxx>@bwbay.ncsa.illinois.edu
- it will ask for your name and email and connect you to Blue Waters
- note: you cannot scp data onto Blue Waters, you can only push/pull data from the login node

RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY
Grab and unpack this exercise

• pull the exercise material from the IU website
  wget http://pages.iu.edu/~hbrunst/ptune15/material/x05.tar.gz

• unpack the material
  tar xzvf x05.tar.gz
Are you ready?

Please, put your answer on

http://m.socrative.com Room: PTUNE15
Matrix multiply

- simple teaching example
  - study effect of loop optimization techniques

\[ C = A \cdot B \quad (A, B, C \in \mathcal{M}(n \times n)) \]
Matrix multiplication with three nested loops (ijk)

\[ c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \]

\[
\begin{bmatrix}
\vdots & \cdots & c_{ij} & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\end{bmatrix}
= 
\begin{bmatrix}
\vdots & \cdots & a_i. & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\end{bmatrix} 
\begin{bmatrix}
\vdots & \cdots & b.j & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
\end{bmatrix}
\]
Intuitive solution in the C programming language

```c
double A[N][N];
double B[N][N];
double C[N][N];
for ( i = 0; i < N; i++ ) {
    for ( j = 0; j < N; j++ ) {
        for ( k = 0; k < N; k++ ) {
            C[i][j] += A[i][k] * B[k][j];
            // C[i*N + j] += A[i*N + k] * B[k*N + j];
        }
    }
}
```
Estimate the sustainable floating point performance

- sustainable, i.e. on large arrays
- consider floating point performance of the core
- consider memory bandwidth
- consider cache
- floating point units per core = 2 mult + 2 add
What floating point performance would you expect?

The peak performance of a single core is ~10 GFLOPS
a) 0.1 GFLOPS
b) 1 GFLOPS
c) 5 GFLOPS
d) 10 GFLOPS

Please, put your answer on

http://m.socrative.com Room: PTUNE15
Get interactive compute allocation

- create interactive session
  qsub -I -X -l nodes=1:ppn=16:xe -l walltime=01:30:00
  Job submitted to account: gjs
  qsub: waiting for job 2093872.nid11293 to start
  qsub: job 2093872.nid11293 ready

- go to directory “x05”
  cd x05
Measure performance of intuitive implementation

- setup environment
  module switch PrgEnv-crays PrgEnv-gnu
- compile matrix multiply example
  cc -O3 -std=c99 matmul0.c -o matmul0
- run matrix multiply example
  aprun -n 1 ./matmul0
What would you do next?

a) take a closer look at the memory access pattern in the loop nest
b) try out different compiler optimization options
c) unroll the loop so that it can be vectorized by the compiler
d) introduce OpenMP pragmas to get more CPU power

Please, put your answer on

http://m.socrative.com Room: PTUNE15
Is there a better loop order than ijk?

a) no, the memory access pattern is optimal this way
b) yes, but the compiler already takes care of it
c) no, the cache ensures that there is no difference
d) yes, ikj would be better

Please, put your answer on

http://m.socrative.com  Room: PTUNE15
Intuitive solution in the C programming language

double A[N][N];
double B[N][N];
double C[N][N];
for ( i = 0; i < N; i++ ) {
    for ( j = 0; j < N; j++ ) {
        for ( k = 0; k < N; k++ ) {
            C[i][j] += A[i][k] * B[k][j];
            // C[i*N + j] += A[i*N + k] * B[k*N + j];
        }
    }
}
Would you use the same loop order in Fortran?

a) yes, Fortran is a similar high level programming language
b) no, Fortran requires dynamically allocated vectors
c) no, Fortran stores matrices column-wise
d) yes, Fortran is well known for its outstanding matmul performance

Please, put your answer on

http://m.socrative.com Room: PTUNE15
Modify the loop order and measure its performance

• change source code with your favorite editor
• compile matrix multiply example
  \[\text{cc } -O3 -std=c99 \text{ matmul-mod1.c } -o \text{ matmul-mod1}\]
• run matrix multiply example
  \[\text{aprun } -n 1 \text{ ./matmul-mod1}\]
Did you make the changes and obtained ~1 GFLOPS?

a) yes
b) no, the compiler complains about the source code
c) no, I still get far less than 1 GFLOPS

Please, put your answer on

http://m.socrative.com Room: PTUNE15
Let’s make some more experiments
for ( uint32_t i = 0; i < dim; i++ )
{
    for ( uint32_t k = 0; k < dim; k++ )
    {
        for ( uint32_t j = 0; j < dim; j++ )
        {
            C[ i * dim + j ] += A[ i * dim + k ] * B[ k * dim + j ];
        }
    }
}

for ( uint32_t i = 0; i < dim; i++ )
{
    for ( uint32_t k = 0; k < dim; k++ )
    {
        double temp_A = A[ i * dim + k ];
        double* temp_B = & B[ k * dim ];
        double* temp_C = & C[ i * dim ];

        for ( uint32_t j = 0; j < dim; j++ )
        {
            // C[i][j] += A[i][k] * B[k][j]
            *temp_C += temp_A * *temp_B;
            temp_B++;
            temp_C++;
        }
    }
}
Unroll inner loop (5)
Why is code much slower than peak performance?

a) the code does not make use of the vector units
b) the code is memory bound
c) the programming language C induces to much overhead
d) the algorithm does not implement enough data locality

Please, put your answer on

http://m.socrative.com Room: PTUNE15
What else can we do?

a) implement blocking of memory accesses
b) use optimized vendor libraries (e.g. mkl)
c) pick another programming language
d) go multithreaded

Please, put your answer on

http://m.socrative.com Room: PTUNE15
https://connect.iu.edu/ptune15

QUESTIONS?