Contents

1 Matrix Multiplication with GCC

1.1 Abstract

Large matrix multiplications can be very handy considering the limitations of cache and the speed of the the RAM access. This report is about conducting matrix multiplication with low level optimizations. Loop tiling and LLVM optimizations are used in order to provide an acceptable result for a better scheduling and work load reduction.

1.2 Basic Matrix Multiplication

The first version of matrix multiplication is done with two arrays of integers(simply for testing). My computer is an Alienware 17 R5, and some of the hardware specifications are listed in Table. 1:

Params	Value
Architecture:	$x86_{64}$
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	8
On-line CPU(s) list:	0-7
Thread(s) per core:	2
Core(s) per socket:	4
Socket(s):	1
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	6
Model:	60
Stepping:	3
CPU MHz:	901.875
BogoMIPS:	4788.98
Virtualization:	VT-x
L1d cache:	32K
L1i cache:	32K
L2 cache:	256K
L3 cache:	6144K
NUMA node0 CPU(s):	0-7

Represented below, is my code for basic matrix multiplication:

```
for(m=0; m<i; ++m){
    for(n=0; n<k; ++n){
        for(l=0; l<j; ++l){
            result[m*k+n] += mat1[m*j+l]*mat2[l*k+n];
        }
    }
}
```