Roofline and Matrix Multiplication PAPI Analysis

José Alves, Rui Brito

Universidade do Minho

Braga, November 2012

Index

Sources for Machine Specifications

Sources used for a complete profile:

Linux System Information (/proc/cpuinfo, /proc/meminfo) To gather specifications on hardware;

Web (ark.intel.com, crucial.com) For micro architecture and memory specifications;

Linux Tools and Packages (dmidecode, sysctl, bandwidth) To gather memory, cpu and bandwidth info;

Machines Specs

Manufacter:	Apple
Model:	MacBook Pro late 2008
Processor	
Manufacturer:	Intel
Arch:	Core
Model:	Core 2 Duo T9600
Cores:	2
Clock Frequency:	2.80 GHz
FP Performance's Peak:	44.8 GFlops/s

Table : MacBook Pro late 2008 specifications

Machines Specs

Cache	
Level:	1
Size:	32KB + 32KB
Line Size:	64 B
Associative:	8-way
Memory Access Bandwidth:	40 GB/s
Level:	2
Size:	6 MB
Line Size:	64 B
Associative:	24-way
RAM	
Type:	SDRAM DDR3 PC3-8500
Frequency:	1067 MHz
Size:	4 GB
Num. Channels:	2
Latency:	13.13 ns

Table : MacBook Pro late 2008 specifications

Machines Specs

Manufacter:	HP
Model:	Pavillion dv6-2190ep
Processor	
Manufacturer:	Intel
Arch:	Nehalem
Model:	i7-720QM
Cores:	4
Clock Frequency:	1.60 GHz
FP Performance's Peak:	51.2 GFlops/s

Table : HP Pavillion dv6-2190ep specifications

Machines Specs

Cache	
Level:	1
Size:	32KB + 32KB
Line Size:	64 B
Associative:	4/8-way
Memory Access Bandwidth:	22 GB/s
Level:	2
Size:	256 KB
Line Size:	64 B
Associative:	8-way
Level:	3
Size:	6 MB
Line Size:	64 B
Associative:	12-way
RAM	
Type:	SDRAM DDR3 PC3-10600
Frequency:	1333 MHz

Problem

Analyse the performance of a **matrix multiplication** algorithm,

$$MatrixA * MatrixB = MatrixC \tag{1}$$

wich contains a triple nested loop with the indexes i,j and k (line,column and position).
The implementation used runs two versions of the problem, one multipying matrixA with matrixB, and another multipying matrixA with the transpose of matrixB.

Algorithm

Standard implementation of a matrix multiplication in C.

```
for (i = 0; i < size; i++) {
    for (j = 0; j < size; j++) {
        for(k = 0; k < size; k++) {
            acc += matrixA[i][k] * matrixB[k][j];
            }
            matrixC[i][j] = acc;
            acc = 0;
        }
    }
```

Counters Used

Used counters gathered by PAPI:

PAPI_TOT_CYC Total cycles;

PAPI_TOT_INS Total instructions

PAPI_LD_INS Load Instructions

PAPI_SR_INS Store Instructions

PAPI_FML_INS Multiply instructions

PAPI_FDV_INS Division instructions

PAPI_VEC_INS Vector Instructions

PAPI_FP_OPS Floating point operations

PAPI_L1_DCA L1 data cache accesses

PAPI_L1_DCM L1 data cache misses

PAPI_L2_DCA L2 data cache accesses

PAPI_L2_DCM L2 data cache misses

Test cases

Test cases were selected to fit on the multiple memory levels.
Each Test case was run 4 times for each version of the problem.

Memory	Size	Matrix Size
L1	30 KB	50
L2	255 KB	146
L3	3 MB	500
RAM	7.68 MB	800

Table : Test cases

Memory Accesses

The following table shows the number of memory accesses, through PAPI readings.

Test	PAPI	Accesses/Inst
L1_1	380844	0.49033
L1_2	258412	0.33270
L2_1	9411279	0.42860
L2_2	6318058	0.33449
L3_1	7761996	0.00885
L3_2	152192	0.00020

Conclusion

- Some difficulties measuring memory ceilings;
- Lack of analysis of output from PAPI;

Roofline and Matrix Multiplication PAPI Analysis

José Alves, Rui Brito

Universidade do Minho

Braga, November 2012