Design of SpeedMark benchmark

Jozo Dujmović

Goals of SpeedMark Benchmark

- Measurement of CPU speed
- Development of a single aggregated indicator of processor speed
- Focusing on CPU resources:
  - processor cores and parallelism of their work
  - efficient organization of machine instructions
  - processor registers
  - main memory
  - cache memory
  - buses (address, data, control)

Workload characterization

- Processor-bound workload consists of three basic components:
  - Integer arithmetic operations
  - Floating point arithmetic operations
  - General operations (present in all programs)
    - Data transfer
    - Shift and rotate
    - Control (jumps, branching, loops), etc.
- It is sufficient to use two fundamental components: integer and floating point

Typical integer and combinatorial operations

- Sorting and searching
- Compiling
- Combinatorial problems:
  - 8 queens
  - Towers of Hanoi
  - Combinatorial games
- Recursive problems:
  - Fibonacci numbers, binomial coefficients
  - Ackermann function
- Integer random number generation

Typical floating point operations

- Function evaluation
- Solving systems of linear equations
- Matrix inversion
- Matrix multiplication
- Computation of integrals
- Differential equations
- Operations with polynomials (e.g. roots)
- Statistical data processing
- Summation of series
- Minimum and maximum of functions

Measurement of processor time

```
#include<time.h> // clock_t clock( void );
double sec(void)
{return double(clock()) /
            double(CLOCKS_PER_SEC);
}

double millisec(void)
{ return clock()*1000. /
            CLOCKS_PER_SEC;
}
```

Measurement of clock resolution

```
T1 = T2 = sec( );
// Wait for clock change
while(T2 == T1) T2 = sec( );
r = T2-T1 ;
```

Measurement of elapsed time

```
double t1,t2;
t1 = seconds();
    MeasuredProgram(parameters);
t2 = seconds();
cout<< "\nTime=" << t2-t1 << " sec";
// To get accurate results use elapsed
// run times of 3 seconds or more
```

The concept of SpeedMark program

- Run 10 seconds the matrix inversion function MINV (or a similar workload)
- Compute the matrix inversion (floating point) speed
- Run 10 seconds the quicksort function QS (or a similar workload)
- Compute quicksort (integer) speed
- Compute and display the average speed in operations/min (operation = MINV or QS)

Measurement of speed

1. Select a constant measurement time, e.g. T = 10 seconds
2. Initialize the operations counter Nop=0 and the start time start=sec()
3. Perform operation, increment Nop, and measure the current time t=sec()
4. If t < start+T repeat the step 3
5. Compute speed V=Nop/(t-start)

Speed measurement program

```
int Nint=0 ;
double START,  Vint;
START = sec( );
while(sec( ) < START+10) {QS( ); Nint++;}
Vint = 60*Nint/(sec( )-START);
//  QS( ) = quicksort function (generate unsorted array, sort, and test the results)
//  Vint = speed measured in quicksorts
//  per minute
```

Computation of SM

- Vint = measured speed of integer operations
- Vfloat = measured speed of floating point operations
- SM = 2 * Vint * Vfloat / (Vint + Vfloat)
- SM uses harmonic means of Vint and Vfloat based on equal weights
- SM can also be defined using different weights SM=1/(W/Vint+(1-W)/Vfloat)

Conclusions

- SpeedMark workload is a mix of typical integer (combinatorial) and typical floating point operations
- SpeedMark benchmark generates a single compound (aggregated) performance indicator (the average processor speed which is also called SpeedMark)
- Parallel execution of multiple SpeedMark programs can be used to measure the effects of parallelism provided by multiple processor cores