PC, FPGA와의 성능 비교 분석을 통한 QEMU의 개선방안 연구

최병준, 서태원
고려대학교 컴퓨터학과
e-mail:cbj5210@nate.com, suhtw@korea.ac.kr

A Study on The Improving Effectiveness of QEMU Based on The Comparative Performance Analysis of PC and FPGA

Byung-Jun Choi, Tae-Weon Suh
Dept of Computer Science and Engineering, Korea University

요 약

본 연구에서는 대표적인 오픈소스 virtual platform인 QEMU와 PC, FPGA에 다양한 운영체제(Windows, Linux, Android, µC/OS-II)를 포팅하고 벤치마크 프로그램을 수행함으로써 성능을 비교 분석하였다. 실험 결과 부동소수점 연산의 성능이 상대적으로 낮게 측정되었으며 이를 토대로 성능 취약점을 분석하고 QEMU를 개선하기 위한 방안을 연구하였다.

1. 서론

QEMU는 다양한 종류의 프로세서(x86, ARM, AMD64, Alpha, MIPS, SparC, PowerPC)와 주변 기기들을 포함한 컴퓨터 시스템 전체를 에뮬레이트 할 수 있는 오픈소스 virtual platform이다. 본 논문에서는 여러 OS를 PC, FPGA(Field Programmable Gate Array), QEMU에 포팅하고 벤치마크 프로그램을 수행함으로써 성능을 비교 분석하고 이를 토대로 QEMU의 개선방안을 찾는 연구를 진행하였다.

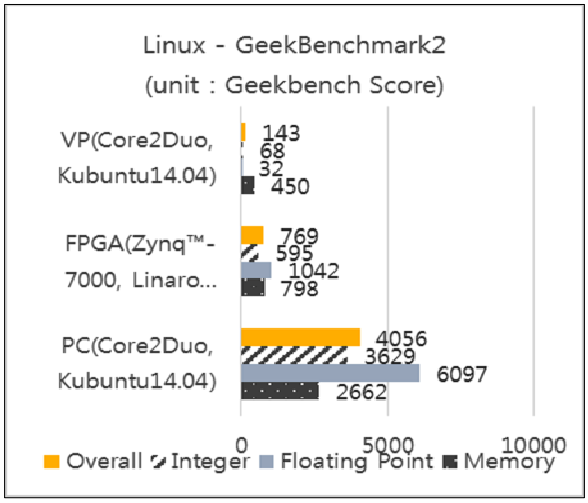
2. 기존 연구

본 연구의 이전 연구인 QEMU를 이용한 Open Source Virtual Platform의 효용성 연구[1]에서는 <표 1>과 같은 실험 환경에서 Linux를 포팅하고 동일한 512MB 메모리를 사용하여 GeekBenchmark2 프로그램을 수행, 성능을 비교 분석하였다. 벤치마크 프로그램 수행 결과는 (그림 1)과 같으며 Geekbench Score는 성능에 비례한 수치를 나타낸다. 가장 큰 격차를 보인 연산은 PC와 FPGA의 경우 정수형(integer) 연산으로 약 6배 정도의 성능 차이를 보였고 FPGA와 VP의 경우 부동소수점 연산으로 약 32.56배의 성능 차이를 보였으며 PC와 VP의 경우 부동소수점 연산에서 약 190.53배의 차이를 보였다. 종합적으로는 PC가

본 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 기본연구지원사업 지원을 받아 수행되었음 (NRF-2014R1A1A2059652)

<표 1> 리눅스 환경에서 PC, FPGA, VP의 실험 환경

	PC	FPGA	VP
Application	Geek Bench2	Geek Bench2	Geek Bench2
OS			Kubuntu-14.04
Virtual Platform			**QEMU** (Intel Core2duo)
Host OS	Kubuntu-14.04	Linaro 12.09	Ubuntu 14.04
Host Machine	Intel Core2duo	ZedBoard (ARM Cortex-A9)	Intel i7 4GB Memory



(그림 1) 각 플랫폼에서의 Geek Benchmark2 Score

FPGA보다 약 5.27배, FPGA가 VP보다 5.38배, PC가 VP보다 약 28.36배 더 좋은 성능을 보였다. 또한 통상적으로 QEMU는 다른 기기에 비해 부동소수점(floating point) 연산에서 좋지 못한 성능을 보였다.

3. OS에 따른 QEMU의 성능 비교 분석

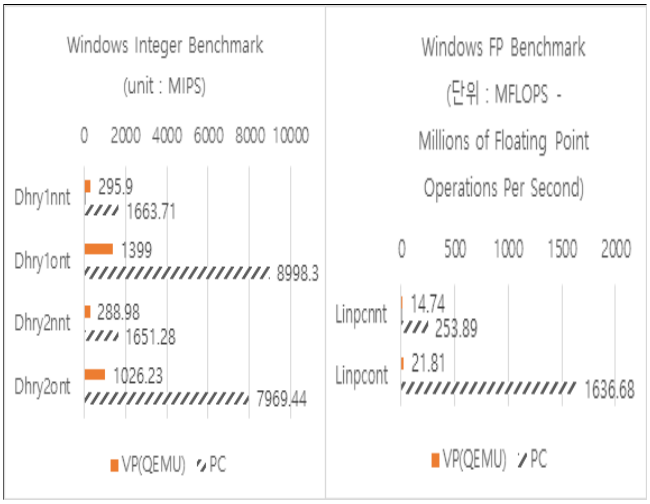
본 실험에서는 QEMU와 PC, FPGA(Zed-board)에 여러 OS를 설치하고, 벤치마크 프로그램을 구동해봄으로써 성능을 비교, 분석하였다.

3.1. Windows

<표 2>의 환경에서 'BenchNT[2]' 정수형 벤치마크 프로그램인 Dhrystone1 nnt(non-optimized), Dhrystone1 ont(optimized), Dhrystone2 nnt, Dhrystone2 ont와 부동소수점 벤치마크 프로그램인 Linpack nnt, Linpack ont을 수행한 결과는 (그림 2)와 같다. 정수형 연산의 경우 대체적으로 VP는 PC의 5~8배 정도 낮은 성능을 보였으나 부동소수점 연산에서는 Linpack nnt의 경우 약 17.22배, Linpack ont의 경우는 약 75.04배 정도 낮은 성능을 보여 VP와 PC의 성능 격차가 크다는 것을 확인할 수 있다.

<표 2> PC, VP 성능분석 실험 환경(Windows)

	PC	VP
Application	BenchNT	BenchNT
OS		Windows Vista
Virtual Platform		QEMU (Intel Core2duo, 512MB memory)
Host OS	Windows Vista	Windows 7
Host Machine	Intel Core2duo 512MB memory	Intel i7 16GB Memory



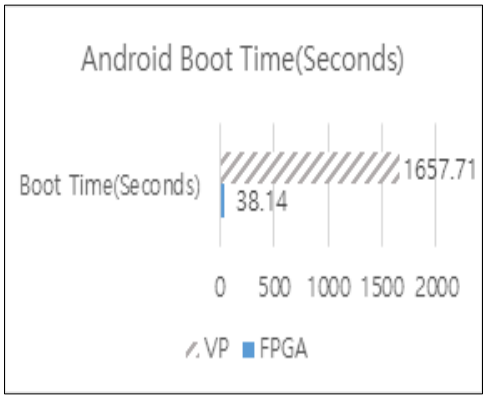
(그림 2) Windows 환경에서 벤치마크 수행 결과

3.2 Android

본 실험에서는 Linaro[3]에서 제공하는 소스를 사용하여 <표 3>의 환경에서 QEMU에 Android를 포팅 하였다. 그러나 QEMU에 가장 최적화가 잘 되어있다고 알려져 있는 Linaro Android 마저도 성능이 좋지 않아 벤치마크 프로그램을 구동하는 것이 현실적으로 불가능 했다. 따라서 다른 OS와는 다르게 Android의 경우 부팅 시작부터 완료까지의 시간을 측정하였다. 그 결과는 (그림 3)과 같으며 FPGA가 VP에 비해 약 43.46배정도 좋은 성능을 보였다.

<표 3> FPGA, VP 성능 분석 실험 환경(Android)

	FPGA	VP
OS		Linaro Android
Virtual Platform		QEMU(ARM vexpress-A9, 512MB Memory)
Host OS	Linaro Android	Ubuntu 14.04
Host Machine	ZedBoard (ARM Cortex-A9, 512MB memory)	Intel i7, 4GB Memory



(그림 3) Android 부팅 시간 측정 결과

3.3 µC/OS-II

µC/OS-II는 가장 대표적인 RTOS(Real Time Operating System)로 다른 OS에 비해 매우 경량화 된 구조와 크기를 가지고 있어 OS의 기능을 모두 사용한다고 하더라도 크기가 30KB를 넘지 않는다. µC/OS-II에서 태스크(task)의 우선순위는 두 개 이상의 태스크가 특정 자원에 대하여 경쟁 상태에 놓여있다면, 어떤 상황이라고 할지라도 둘 중에서 우선순위가 높은 쪽이 항상 그 자원을 독점하도록 되어있다. 또한 round robin을 지원하지 않기 때문에 같은 우선순위를 가진 태스크가 두 개 이상 존재할 수도 없다.[4]

본 실험에서는 µC/OS-II를 FPGA와 QEMU에 포팅하고 태스크를 수행하여 성능을 측정하였다. 실험환경은 <표 4>와 같으며, 태스크는 원주율(π, 3.14159265358979…)을 무한급수를 이용한 라이프니츠의 공식을 사용하여 소수점

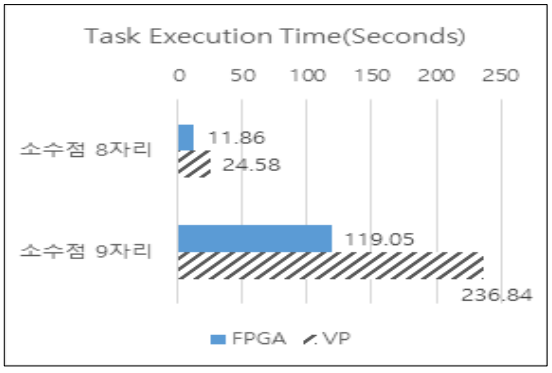
8, 9자리까지 구하는 것으로 작성하였다. 소수점 8자리까지 구하기 위해서는 <표 5>의 소스코드 내부의 While 반복문이 99115148번 수행되고, 9자리까지 구하기 위해서는 반복문이 919102060번 수행 되어야 한다. 태스크 수행 결과는 (그림 4)와 같으며 대략적으로 FPGA가 VP보다 2배정도 좋은 성능을 보였다.

<표 4> FPGA, VP 성능분석 실험 환경(µC/OS-II)

	FPGA	VP
Application	Task	Task
OS		µC/OS-II
Virtual Platform		QEMU (Intel Core2duo, 512MB memory)
Host OS	µC/OS-II	
Host Machine	ZedBoard (ARM Cortex-A9, 512MB memory)	

<표 5> µC/OS-II에서 수행한 태스크(원주율 계산)

```
int main(void){
   double pi = 3.1415926535;
   double tolerance=0.000000001;
   double sum = 0;
   long int k = 0;
   double error;
   error = pi - sum;
      while ( error > tolerance || error < -tolerance ){
         ++k;
         if ( k % 2 ) sum += 4.0 / (2.0 * k - 1.0);
         else sum -= 4.0 / (2.0 * k - 1.0);
         error = pi - sum;
      }
          printf("After %ld terms: \n", k);
          printf("pi = %.15f\n", sum);
}
```



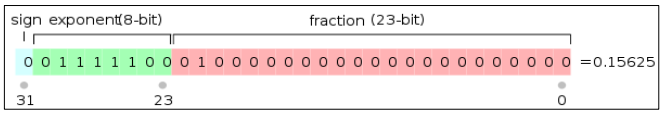
(그림 4) FPGA, VP 성능 분석 결과(µC/OS-II)

4. QEMU의 개선방안

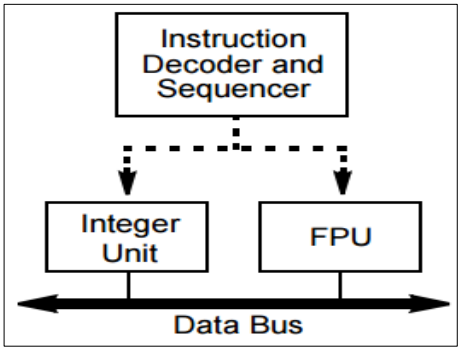
본 논문의 실험들을 통해 QEMU의 성능을 파악한 결과 QEMU의 성능을 1이라도 보았을 때 각 OS별로 PC와 FPGA의 상대적인 성능을 나타낸 수치는 <표 6>과 같다. 이를 토대로 QEMU가 PC나 FPGA에 비해 좋은 성능을 보이지 못하는 이유는 다음과 같은 관점에서 살펴볼 수 있다. 첫째 하드웨어(PC, FPGA)의 처리속도가 소프트웨어(QEMU)의 처리속도보다 빠르기 때문이다. 둘째 QEMU는 머신 코드를 생성하기 위해 타겟 바이너리에서 수행되어질 명령어 목록(Translation Block)을 선택하여 중간 코드(Intermediate Representation)로 번역하고 중간 코드를 호스트 CPU의 명령어로 번역하고 레지스터 할당을 하는 dynamic translation을 수행한다. 하지만 QEMU는 dynamic translation에서 쓸모없이 수행될 반복적인 코드를 생성하기 때문에[5] PC나 FPGA에 비해 동일한 프로그램을 실행하더라도 수행해야 할 명령어가 상대적으로 많아 좋은 성능을 보이기 어렵다. 마지막으로 본 논문의 모든 실험 결과에서 확인할 수 있듯이, QEMU는 어떤 OS를 포팅 하더라도 공통적으로 다른 연산에 비해 부동소수점 연산에서 좋은 성능을 보이지 못했다. 그 이유를 찾기 위해 QEMU의 소스코드를 분석한 결과 다음과 같은 결론을 얻을 수 있었다. IU(Integer Unit) 혹은 ALU(Arithmetic Logic Unit)를 사용하는 정수형 연산과는 다르게 부동소수점 연산은 일반적으로 (그림 5)의 IEEE 754 표준에 따른 FPU(Floating Point Unit) 하드웨어 논리회로에서 (그림 6)처럼 별도로 수행된다. 하지만 QEMU에서의 부동소수점 표현은 (그림 7)처럼 IEEE 754 표준을 그대로 구현하긴 하였으나 부동소수점 연산을 정수형 변수로 치환하여 수행하기 때문에 하나의 부동소수점 연산을 수행할 때마다 여러 개의 정수형 연산을 수행해야 한다. 즉 상대적으로 처리해야할 명령어의 수가 PC나 FPGA에 비해 많아질 수밖에 없으므로 OS나 벤치마크 프로그램의 종류와 상관없이 QEMU의 부동소수점 연산은 다른 연산에 비해 좋은 성능을 보일수가 없다. 따라서 dynamic translation의 효율성을 높이고 부동소수점 연산의 수행 방식을 변화시키는 것이 QEMU 성능 개선방안의 하나라고 판단된다.

<표 6> 각 기기별 OS에 따른 성능 비교표

Device / OS	PC	FPGA	QEMU
Windows	19.6	X	1
Linux	28.3	5.3	1
Android	X	43.4	1
µC/OS-II	X	2	1



(그림 5) IEEE 754(32-bit floating point standard) 예시[6]



(그림 6) 정수형 장치(Integer Unit)와 부동소수점 장치(FPU)의 관계[7]

```
// QEMU/target-i386/machine.c

union x86_longdouble
{
    uint64_t mant;
    uint16_t exp;
};


// QEMU/include/fpu/softfloat.h

typedef uint16_t float16;
typedef uint32_t float32;
typedef uint64_t float64;


// QEMU/fpu/softfloat.c

static float32 addFloat32Sigs
(float32 a, float32 b, flag zSign, float_status *status)
{
    int_fast16_t aExp, bExp, zExp;
    uint32_t aSig, bSig, zSig;
    int_fast16_t expDiff;

    aSig = extractFloat32Frac( a );
    aExp = extractFloat32Exp( a );
    bSig = extractFloat32Frac( b );
    bExp = extractFloat32Exp( b );
}
```

(그림 7) QEMU 소스코드(부동소수점)

5. 결론

본 논문은 QEMU를 사례로 오픈소스 virtual platform의 성능을 PC, FPGA와의 비교를 통해 분석하였다. 성능 평가를 위해 Windows, Linux, Android, µC/OS-II를 포팅하였고 BenchNT, GeekBench2 등의 벤치마크 프로그램을 사용하였다. 성능 측정 결과 QEMU는 Windows 환경에서 PC보다 19.6배, Linux 환경에서 PC보다 28.3배, FPGA보다 5.3배 낮은 성능을 보였다. 또한 Android 환경에서는 OS의 소스코드의 최적화가 이루어지지 않는 등의 문제로 FPGA보다 43.4배 낮은 성능을 보였으며 µC/OS-II 환경에서는 FPGA보다 2배 느린 태스크 처리 성능을 보였다. 모든 OS 환경에서 공통적으로 QEMU는 부동소수점 연산에서 좋지 못한 성능을 보였으며 이를 토대로 QEMU의 부동소수점 연산 방식을 조사한 결과 부동소수점 연산을 정수형 연산으로 치환하여 수행하는 것을 확인할 수 있었다. 본 논문에서 수행한 QEMU의 실측을 통한 성능 평가는 향후 virtual platform의 성능개선 및 올바른 방향 설정을 위한 객관적인 자료를 제공한다.

참고문헌

[1] 최병준, and 서태원. "QEMU를 이용한 Open Source Virtual Platform의 효용성 연구." 한국정보처리학회 추계학술발표대회 논문집 제22권 제2호(2015.10): 76-78.

[2] http://www.roylongbottom.org.uk

[3] http://www.linaro.org

[4] https://en.wikipedia.org/wiki/Micro-Controller_Operating_Systems_(MicroC/OS)

[5] 홍성현, 정동헌, and 문수묵. "QEMU 를 이용한 ARM 플랫폼에서 x86 바이너리 수행의 성능 평가 분석." 대한전자공학회 2010 년 정기총회 및 추계종합학술대회 (2010): 476-477.

[6] https://en.wikipedia.org/wiki/IEEE_754-1985

[7] Intel Architecture Software Developer's Manual (Floating-Point Unit)