동시 다중 쓰레딩을 이용하는 마이크로 프로세서 성능평가

이정훈[O*] 오영은[*] 박형우[**] 김진석[*]
서울시립대학교 컴퓨터과학부[*], 한국과학기술연구원 슈퍼컴퓨팅센터[**]
{jhlee94, iceize, jskim}@venus.uos.ac.kr[*], hwpark@hpcnet.ne.kr[**]

Performance Evaluation of a Simultaneous MultiThreading

Jung-Hoon Lee[O*], Young-Eun Oh[*], Hyoung-Woo Park[**] and Jin Suk Kim[*]
School of Computer Science, University of Seoul[*] and Supercomputing Center, KISTI[**]

요 약

프로세서의 효율을 높이기 위한 방법으로 독립적인 쓰레드들을 한 프로세서 사이클에 동시에 실행시킬 수 있는 SMT 기술에 관한 많은 연구가 수행되어왔다[1, 2, 3, 4]. 많은 연구에서 SMT 기술에 대한 성능을 시뮬레이션 수준에서 측정하였기 때문에, 실제 환경에서 SMT 기술의 성능을 측정할 필요가 있다. 본 논문에서는 SMT 기술이 구현된 프로세서에서 각종 벤치마킹을 직접 수행해 봄으로써 실제 환경에서의 성능을 측정해 보았으며, 이를 기존의 SMP와의 비교를 통해 SMT 기술이 실제로 얼마만큼 좋은 성능을 낼 수 있는지 실험을 통해 보였다.

1. 서 론

프로세서의 효율을 높이기 위한 방법으로 다양한 연구가 진행되고 있으며, 그 예의 하나가 SMT(Simultaneous MultiThreading)[1] 이다. SMT는 각각의 프로세서 사이클에서 가용한 프로세서 자원들을 공유하여 여러개의 쓰레드들을 동시에 실행시킴으로서 프로세서의 효율을 높이게 된다[4]. SMT는 응용프로그램의 관점에서 하나의 프로세서가 마치 다수의 프로세서인 것처럼 보이게 한다. 또한 동시에 여러개의 독립적인 쓰레드를 수행시켜 프로세서의 병렬성을 극대화하고 결과적으로 전체적인 프로세서 성능의 증가를 이루게 된다. 하지만 많은 연구에서 SMT의 성능을 시뮬레이션 수준에서만 측정하였으며, 따라서 실제 환경에서 SMT 기술의 성능을 측정할 필요가 있다. 최근 인텔에서는 하이퍼쓰레딩이라는 이름으로 SMT 기술을 구현한 프로세서들을 출시하였다. 본 논문에서는 인텔의 SMT 기술을 지원하는 프로세서상에서 각종 벤치마킹을 수행해 봄으로서 SMT 성능을 실제적으로 측정해 보았으며, 이를 기존의 SMP(Symmetric MultiProcessing)와의 비교를 통해 SMT 기술이 실제로 얼마만큼 좋은 성능을 낼 수 있는지 실험을 통해 보이고 분석하였다.

2. 관련연구

프로세서의 성능을 높이기 위한 연구는 크게 ILP(Instruction-Level Parallelism)와 TLP(Thread-Level Parallelism)를 증가시키는 방식으로 나눌 수 있다. ILP는 한 프로세서 사이클상에서 실행시킬 수 있는 인스트럭션의 개수를 증가시키고, TLP는 다수의 프로세서에 여러개의 쓰레드를 동시에 수행시키는 접근방법을 취하고 있다. 하지만 근래에 들어와서 ILP와 TLP를 모두 증가시킴으로서 프로세서의 낭비되는 자원들을 효율적으로 활용하고 궁극적으로 전체적인 프로세서 성능을 높일 수 있는 SMT 기술이 제안되었다. SMT 기술은 하나의 물리적인 프로세서가 다수의 논리적인 프로세서로 구성됨으로서 물리적인 프로세서 자원들을 공유하여 한 프로세서 사이클당 여러개의 쓰레드를 동시에 수행시키는 방법으로 프로세서 성능의 향상을 꾀하고 있다. 이는 운영체제나 사용자 프로세스의 관점에서 다수의 쓰레드를 동시에 실행시킬수 있는 프로세서의 개수가 증가되었음을 의미하며, 마이크로 아키텍쳐의 관점에서는 물리적인 프로세서 자원들을 공유함으로서 프로세서 자원활용의 효율성을 극대화함을 의미한다.

3. 동시 다중 쓰레딩 기법

동시 다중 쓰레딩 기법은 wide-issue superscalar와 *multithreaded processor*를 혼합한 형태로 구성된다. 즉, superscalar처럼 각각의 사이클에 다수의 인스트럭션을 실행시킬 수 있으며, multithreaded processor처럼 다중 쓰레딩 기법을 지원한다. 이를 도식화하면 그림 1과 같다. superscalar 는

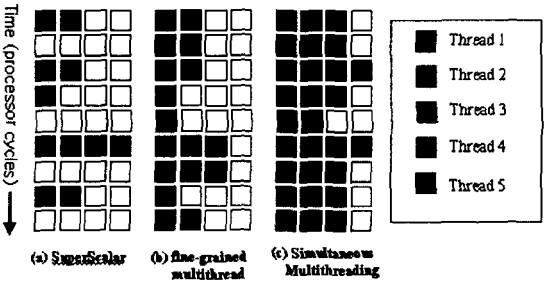

그림 1. 동시다중 쓰레딩 기법[4]

한 사이클당 여러개의 인스트럭션을 수행할수 있음으로서 ILP를 증가시켰고, fine-grained multithread 또는 Traditional Multithreaded 프로세서는 여러개의 하드웨어 문맥을 가지게 되어 여러 사이클동안에 걸쳐서 서로 다른 쓰레드를 독립적으로 처리할수 있음으로서 TLP를 증가시켰다. 하지만 SMT 에서는 여러개의 하드웨어 문맥을 동시에 처리할수 있도록 설계되어 TLP 와 ILP를 모두 증가시켰으며, Vertical issue slot waste 와 horizontal issue slot waste를 동시에 감소시켰다.

4. 실험환경 및 결과

실제 환경에서 SMT 성능을 측정하기 위하여 인텔의 Xeon

MP 1.4 GHz Dual 프로세서를 이용하였다. 컴파일러는 RedHat 패치가 이루어진 gcc 2.96 버전을 사용하였으며, 운영체제로는 Microsoft Windows XP Professional (Service Pack 1) 과 Linux 2.4.18 커널을 이용하였다. 또한 SMT 성능을 SMP와 좀 더 정확하게 비교하기 위하여 SMT 머신상에서 하이퍼쓰레딩 기능을 제거함으로서 단순히 기존의 SMP 머신으로 동작하게 하여 같은 방법으로 성능을 비교 측정하였다. 성능 벤치마킹을 수행할 때에 다양한 측면의 성능을 비교해보기 위하여 한가지 벤치마킹 툴킷에 의존하지 않고 여러 가지 형태의 벤치마킹을 수행하였다. 본 논문에서 수행한 벤치마킹의 방법은 크게 CPU 의 정수 또는 부동소수점 연산수행능력 등의 특정한 형태의 성능을 측정하기 위한 Synthetic 벤치마킹 방법과 시스템의 모든 컴퍼넌트들을 조합하여 실제 응용프로그램을 실행해 봄으로서 전체적인 성능을 측정하기 위한 Application 벤치마킹 방법으로 나뉘어질 수 있다. Synthetic 벤치마킹을 위하여 LMBench[6], UNIX Bench[7], BYTE Bench[8]등의 툴킷을 사용하였으며, Application 벤치마킹을 위하여 pThread 응용프로그램과 Apache 웹서버, 그리고 리눅스 커널 컴파일등의 수행능력을 측정, 비교하였다.

4.1. Synthetic 벤치마킹 실험 결과

Synthetic 벤치마킹 실험에서는 Int32, Int64, Float, Double 형태의 연산 수행시 병렬성과 특정 작업 수행시 병행성을 변화시킴으로서 발생하는 성능의 차이, 그리고 다양한 특성을 가진 작업에 따른 성능의 변화를 비교 측정하였다.

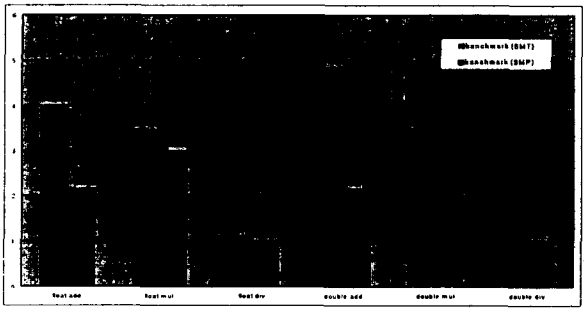

그림 2 Float, Double 연산 수행시 병렬성

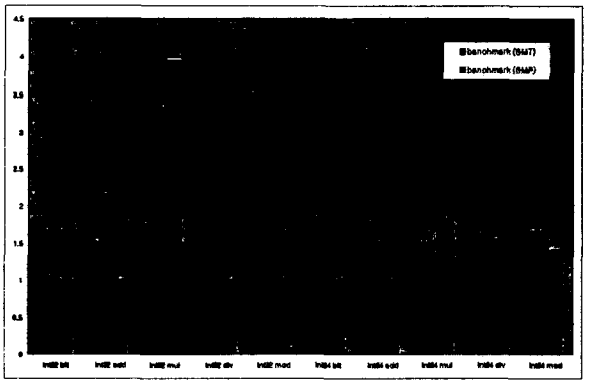

그림 3 Int32, Int64 연산수행시 병렬성

<그림 2>와 <그림 3>은 Int32, Int64, Float, Double 형태의 각종 연산 수행시 병렬성을 LMBench 툴킷을 사용하여 측정한 결과이다. 대부분의 연산에서 SMT가 SMP보다 우수하며, 특히 Int 형태의 연산보다 Float, Double 형태의 연산 수행시 병렬성이 더 높아졌음을 보여주고 있다. 일반적으로 프로세서의 개수가 증가함에 따라 병렬성도 함께 증가하기 때문에 논리적인 프로세서의 개수가 증가되어 있는 SMT가 SMP보다 우수함을 알 수 있다.

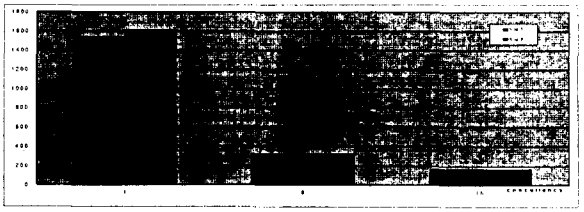

그림 4. 병행성 변화에 따른 SMT와 SMP의 성능비교

<그림 4>는 병행성 변화에 따른 SMT와 SMP의 성능의 차이를 UNIX Bench 툴킷을 사용하여 비교한 결과이다. 병행성이란 어떠한 작업을 독립적으로 동시에 처리하는 것을 의미하며, 즉 1개의 프로세스가 처리하는데 1600초가 걸리는 독립적인 작업을 8개의 프로세스가 동시에 처리하게 된다면 약 200초안에 마칠수 있게 된다. 본 실험에서는 병행성이 1, 8, 16으로 변화함에 따라 작업 수행완료 시간을 초단위로 비교 측정하였다. 그림에서 세로축이 병행성 변화에 따른 작업 수행완료 시간을 나타내며 작을수록 높은 성능을 나타낸다. 이 실험 결과를 통하여 SMT, SMP 모두 병행성이 증가함에 따라 상대적으로 성능 또한 증가됨을 알 수 있으며, 전체적으로 SMT가 SMP보다 우수함을 보여주고 있다.

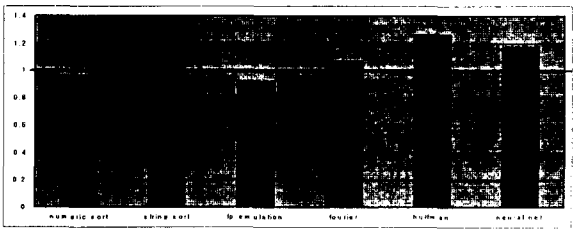

그림 5 다양한 작업에 따른 SMP에 대한 SMT의 상대 성능

<그림 5>는 BYTE Bench를 이용하여 다양한 작업을 수행시킴으로서 측정되는 SMP에 대한 SMT의 상대 성능을 측정한 결과이다. 즉, SMP의 성능을 1로 표현할 때에 SMT의 상대적인 성능을 나타내며, 대부분의 작업에서 SMT의 성능이 우수하고, 특히 Huffman, Neural net 과 같은 Float, Double 형태의 연산이 상대적으로 많이 포함되어 있는 작업의 경우 더욱 높은 성능을 보여주고 있음을 알 수 있다.

4.2. Application 벤치마킹 실험 결과

Application 벤치마킹 실험에서는 pThread[9] 응용프로그램과 Apache 웹서버, 그리고 리눅스 커널 컴파일 수행시의 성능을 측정하여 비교하였다. Application 벤치마킹은 시스템을 구성하는 여러 컴퍼넌트들의 고유의 성능들이 유기적으로 관련되어 전체적인 성능으로서 측정된다. 따라서 정확한 비교를 위하여 같은 시스템상에서 SMT 기능을 수행하도록 설정하여 성능을 측정한 후, SMT 기능을 수행하지 않는 단순한 SMP 시스템으로서 동작하도록 설정하여 측정한 성능값을 비교하는 방법으로 실험하였다.

```
a> Main Routine
  #include <stddef.h>
  #include <stdio.h>
  #include <unistd.h>
  #include <pthread.h>
  #define MAX 200
  int main (void)
  {
     int i;
     pthread_t th[MAX];
     void *retval;

     for (i = 0; i < MAX; i++)
        pthread_create (&th[i], NULL, process, (void *)i);
     for (i = 0; i < MAX; i++)
        pthread_join (th[i], &retval);

     return 0;
  }

b> Thread Routine
  static void *process (void *arg)
  {
     double i = 0;
     int sum = 0;

     while (1)
     {
        if (i > 1000) break;
        if ((int)i % 2 == 0) sum++;
        i = i + 0.01;
     }

     return NULL;
  }
```

그림 6 pThread 테스트 프로그램

<그림 6>은 pThread 응용프로그램 벤치마킹 실험을 위하여 작성한 루틴이다. MAX 값은 생성할 쓰레드의 개수를 정의하며 이 값에 따라 생성된 쓰레드들이 각각 Thread Routine에서 10만번의 루프를 통해 1씩 더하는 단순 연산들을 수행하게 된다. 이 루틴을 실행하여 측정한 성능 결과는 <그림 7>에서 보여주고 있다.

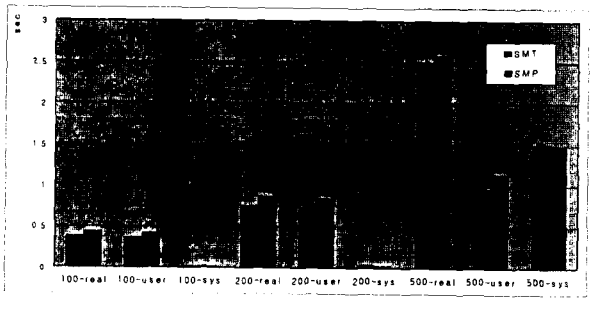

그림 7 pThread 벤치마킹 실험 결과

<그림 7>에서는 MAX값의 변화에 따라 실제 프로그램 수행 완료시간, 프로그램 수행시 운영체제가 사용자 영역에서 수행한 시간, 그리고 커널 영역에서 수행한 시간을 초단위로 측정한 결과값을 도식화한 것이다. 일반적으로 시스템에서 동시에 수행되는 쓰레드가 많아질수록 시스템의 성능이 떨어지며, 대부분의 경우에서 SMT가 SMP보다 성능이 비슷하거나 또는 우수함을 알 수 있다. <그림 8>에서는 Apache 웹서버가 총 2000번의 요청을 동시접속자수를 달리하면서 처리하는 수행시간에 대한 측정값이다. 동시접속자수가 많아질수록 한 접속에서 요청하는 요청수가 작아지므로 상대적으로 요청처리시간이 작아지며, SMT가 SMP보다 좋은 성능을 나타내주고 있음을 알 수 있다. 마지막으로 <그림 9>에서는 리눅스 커널 2.5.49 버전을 컴파일 하였을때의 전체수행시간, 커널영역에서의 수행시간, 그리고 사용자영역에서의 수행시간을 초단위로 측정하여 보여주고 있다. 전체수행시간에서 SMT가 SMP보다 좋은 성능을 보여주고 있으나 커널영역에서 좀더 많은 시간을 소비하고 있음을 알 수 있다.

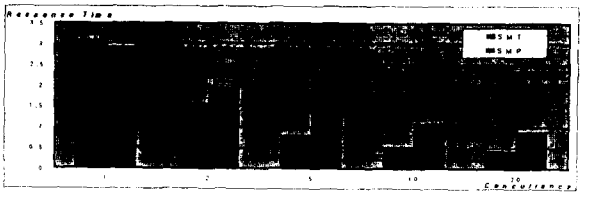

그림 8 동시접속자수의 변화에 따른
Apache 웹서버 요청처리시간

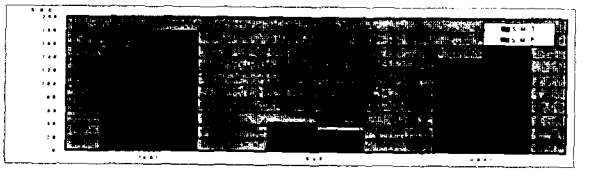

그림 9 리눅스커널 컴파일 수행시간

5. 결론

본 논문에서는 물리적으로 1개의 프로세서가 논리적으로 다수의 프로세서 역할을 수행함으로서 좀더 높은 성능을 이루고자 설계된 SMT를 실제 환경에서 실험하여 기존의 SMP보다 얼마만큼의 좋은 성능을 나타낼 수 있는지 실험을 통해 보였다. Synthetic 벤치마킹 실험을 통하여 전체적으로 SMT가 정수형태의 연산보다 부동소수점 형태의 연산에서 높은 성능을 나타내며, 병행성이 증가함에 따라 좀더 높은 성능을 나타냄을 알 수 있다. 또한 Application 벤치마킹 실험을 통하여 실제 완전한 형태의 시스템에서 전반적으로 SMT가 SMP보다 높은 성능을 보여주고 있음을 알 수 있다.

참고문헌

[1] D. Tullsen. S. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. of the 22rd Annual International Symposium on Computer Architecture, pp. 392-403, June, 1995.*

[2] A. Agarwal, "Performance tradeoffs in multithreaded processors," *IEEE transactions on parallel and distributed systems, September*, 1992.

[3] D. Tullsen, S. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," *Proc. of the 23th Annial International Symposium on Computer Architecture, May*, 1996.

[4] D. Tullsen et al, "Simultaneous Multithreading : a Platform for Next-Generation Processors," *IEEE Micro, pp. 12-19, Oct, 1997.*

[5] 이정훈, 오영은, 김정석, "Benchmark on SMT and SMP," *Technical Report, 서울시립대학교 컴퓨터통계학과, Dec, 2002.*

[6] LM Bench, http://www.bitmover.com/lmbench/

[7] UNIX Bench, http://www.tux.org/pub/tux/benchmarks/System/unixbench

[8] BYTE Bench, http://www.tux.org/~mayer/linux/bmark.html

[9] IEEE Std. 1003.1c-1995, http://standards.ieee.org/reading/ieee/interp/1003-1c-95_int/index.html