병렬 시스템의 최적 프로세서 수 결정

이용욱[0] R.S.Ramakrishna
광주 과학 기술원
motorhead@kebi.com[0], rsr@kjist.ac.kr

Finding Optimal Number of Processors in Parallel System

Lee, YoungUk[0] R.S.Ramakrishna
Dept. of Information & Communication, Kwangju Institute of Science & Technology

요 약

본 연구는 주어진 병렬 프로그램에 대해서 병렬 시스템의 성능을 효과적으로 이끌어낼 수 있는 최적 프로세서 수를 결정하는 방법에 관한 것이다. 최적 프로세서 수는 병렬 시스템의 성능 뿐만 아니라 비용도 같이 고려하여 이들 조건이 균형을 이루어 가장 효율적인 성능을 이끌어낼 수 있는 병렬 시스템의 프로세서 수로 정의된다. 최적 프로세서 수는 주어진 병렬 프로그램에 대해서 얼마나 많은 프로세서를 사용할 수 있는지 알려줄 수 있으므로, 보다 객관적이고 구체적인 병렬 시스템의 확장성 기준으로도 사용될 수 있다.

1. 서 론

현재의 컴퓨터 연산 능력은 이전에는 일찍이 없었던 막강한 기능을 제공하고 있다. 1 GHZ의 속도를 훨씬 웃도는 연산 처리 장치와 수십 기가 바이트 의 저장 능력을 가진 개인용 컴퓨터들이 일상적으로 사용되고 있다. 자리를 잡고 있다. 하지만 그와 비례하여 더욱 막대한 양의 연산을 필요로 하는 문제들이 더욱 많이 요구되고 있다. 현재 붐을 이루고 있는 생명 정보학이 그러한 예라고 볼 수 있을 것이다. 병렬화는 오래 전부터 막대한 양의 연산을 필요로 하는 문제를 빠르고 효율적으로 해결하는 가장 일반적인 방법으로 인식되어 왔다. 복잡한 알고리즘을 사용해야 한다는 단점에도 불구하고 기존의 컴퓨터 시스템들을 이용하여 높은 성능과 효율을 낼 수 있다는 장점때문에 그 활용 범위는 점점 더 넓어지고 있다. 병렬화 하는데 있어서 가장 큰 관심 거리는 과연 얼마나 성능을 높일 수 있는가 하는 것이다. 그리고 '얼마나 많은 프로세서들을 사용해야 가장 높은 성능을 얻을 수 있겠는가?' 라는 질문은 늘 병렬화에 꼬리표처럼 붙어 다니는 고민거리이다.

병렬화의 특성상 연산에 참여하는 프로세서의 수가 증가할수록 시스템 간의 통신 병목 현상이 가중되기 때문에 무한정 프로세서의 수를 늘려 병렬화의 이득을 꾀할 수는 없다. 따라서 병렬화에 참여하는 가장 적절한 시스템의 수를 결정하여 최대의 효율화를 꾀하는 구체적인 방법이 필요하다. 이 적절한 시스템의 수를 최적 프로세서 수라고 한다. 최적 프로세서 수는 주어진 병렬 프로그램을 최고의 성능으로 처리할 수 있게 하는 병렬 시스템의 크기로 정의된다. 그림 1은 가상적인 병렬 프로그램의 처리 시간을 나타낸 모의 실험 결과이다. 일반적인 병렬 프로그램의 연산 시간은 프로세서의 개수가 증가함에 따라 지수적으로 급격하게 감소 하는 반면 통신 시간은 로그 급수로 늘어가는 양상을 보인다. 최적의 프로세서 수는 연산 시간을 최소로 하는 프로세서의 개수 165로 정할 수 있다. Hwang과 Xu는 [1] 통신 시간이 연산 시간보다 더 커지는 순간의 프로세서 개수(그림 1에서 65)를 최적 프로세서 수로 정의하고 있지만 통신 시간이 연산 시간 보다 늘어나도 전체 시간은 계속 감소하고 있음을 확인할 수 있다. 하지만 134를 최적의 프로세서로 결정하는 데도 문제가 있다. 134개의 프로세서를 사용하면 명백하게 연

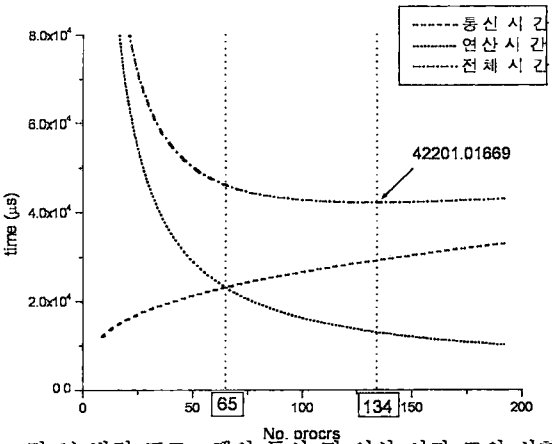

그림 1) 병렬 프로그램의 통신 및 연산 시간 모의 실험 결과

산 속도를 최소로 하여 병렬 시스템의 성능을 최고로 얻을 수 있지만 프로세서 하나를 추가하여 얻을 수 있는 이득의 양이 너무 작다. 즉, 가격대 성능비가 너무 떨어지는 그림.1의 경우 100개의 프로세서를 사용할 경우와 134개의 프로세서를 사용할 경우 얻을 수 있는 시간 이득의 차이는 겨우 1.5% 미만을 나타내고 있다. 따라서 병렬 시스템의 성능을 최고로 이끌어 낼 수 있는 가장 작은 수의 프로세서의를 결정하는 방법이 요구된다.

2. 최적 프로세서 수

본 연구에서는 비용과 연산 성능의 균형을 맞출 수 있는 최적의 프로세서를 찾는 방법을 제안하였다. 구체적인 프로세서의 개수를 결정하는 '최적 효율 기준 [2]'이 제안 되긴 했지만 이 평가 방법은 잘 사용되지 않는 비용(Cost)과 유효성(Effectiveness) 기준을 사용하였다. 하지만 대부분의 확장성 평가 기준은 일반적으로 속도 향상 비율(Speed Up)과 효율성(Efficiency) 기준을 사용하므로 제안된 최적의 프로세서 개수도 이들을 바탕으로 결정하도록 고려하였다.

병렬 시스템의 비용과 성능 사이에 균형을 맞추는 기준은 p개의 프로세서를 사용하는 경우의 효율 $E(p)$ 와 p+1개의 프로세서를 사용할 때의 효율 $E(p+1)$ 의 비율이다.

$$\frac{E(p)}{E(p+1)} \le C \qquad (1)$$

여기서 상수 C는 최대 한계 효율비로 시스템에서 사용할 최대의 허용 가능 효율비를 나타낸다. 식(1)은 프로세서의 수가 하나씩 증가한 비가 이전의 추가할 때마다 효율의 증가는 C를 넘어야 한다는 것을 말한다. 사용할효율 $E(p)$은 속도 향상 비율 $S(p)$ 와 사용 프로세서 수 p 의 곱으로 나타낼 수 있으므로[3], 식 (1)은 다음과 같다.

$$C \cdot S(p+1) - S(p) \ge 0 \qquad (2)$$

최적 프로세서 수 P_{opt} 는 식 (2)를 만족하는 최대의 프로세서 p로 정의한다.

$$P_{opt} = \max_{p}(C \cdot S(p+1) - S(p) \ge 0) \quad (3)$$

제안된 최적 프로세서 수는 주어진 병렬 프로그램을 최고의 효율로 실행시키기 위해서 얼마나 많은 프로세서가 필요한지를 알려주기 때문에 또 다른 확장성 기준으로도 사용될 수 있다. 이것은 기존의 IsoEfficiency[4]나 IsoSpeed[5] 확장성 기준과는 다른 몇 가지 장점을 가지고 있다.

첫째, 기존의 확장성 기준은 주로 병렬 시스템과 프로그램을 상대적으로 비교하여 확장성을 0에서 1사이의 다소 모호한 값으로 제시한다. 반면, 제안된 최적 프로세서 개수 방법은 병렬 시스템이 적당한 성능을 내기 위해 최대로 사용할 수 있는 프로세서의 개수를 구체적으로 제시함으로써 사용자에게 이해하기 쉽고 명확한 기준으로 사용될 수 있다. 둘째, 기존의 확장성 기준들이 오직 고정 시간 형태의 문제들만을 고려한 반면 최적 프로세서 개수는 모든 형태의 문제 형태들에 적용될 수 있다. 문제 크기가 고정되어 있으면 그 문제를 해결하기 위한 적절한 프로세서의 개수를 알려주고, 프로세서의 개수가 고정되어 있는 경우에는 이 것을 최적 프로세서로 하는 문제 크기를 정할 수 있다.

3. 통신 지연 현상 모형화

대상이 되는 병렬 시스템의 통신 지연과 연산 시간을 모형화 할 수 있으면 실험을 하지않고도 최적 프로세서 수를 해석적인 방법으로 구할 수 있다. 연산 시간은 일부 연산을 측정하여 적절한 상수를 곱함으로써 전체 연산 시간 근사할 수 있지만 통신 지연은 시스템마다 다른 특징을 가지고 있으므로 각 시스템 별로 이를 측정하여야 한다. 하지만 이를 위해서 모든 경우에 대하여 각각을 실험하는 일은 거의 불가능한 일이므로 적절한 통신 지연 현상을 모형화하는 것이 필요하다.

통신 지연을 모형화 하는 여러 가지 방법이 제안되었지만 [1,6]은 통신 지연을 측정하고 이를 최소 자승법을 사용하여 하나의 근사식으로 나타내는 실험과 해석의 두 가지 방법을 모두 채택하므로 훨씬 체계적이고 신뢰성있는 통신 지연 모형을 얻을 수 있다. 집합적(collective) 통신의 통신 지연은 전달하는 메시지의 바이트 수와 사용 프로세서 수를 변수로 하는 비선형 함수이다. 통신 지연 시간은 전달하는 메시지간의 바이트 수에는 선형적으로 비례하고 사용하는 프로세서 수에는 로그함수에 비례하는 특성을 가진다 [1].

$$T_{comm} = (t1 + \frac{m}{r_{\infty}}) \log p \qquad (4)$$

식 (4)는 [1]에서 지연된 통신 지연 함수이고 이를 이용하여 측정된 자료의 최소 자승법을 시행하였다. 여기서 $t1$은 시작 시간으로 통신을 하는데 기본적으로 필요한 시간을 나타내며, r_{∞} 은 최대 통신 대역폭으로 무한히 많은 메시지를 보낼 때 사용할 수 있는 통신 대역폭을 나타낸다. m 과 p 는 각각 메시지 바이트 수와 프로세서의 수이다.

통신 지연 모형화의 대상으로 삼은 병렬 시스템은 삼성 종합 기술원의 알파 클러스터 Alpha11이다. 연구의 모든 실험은 Alpha11에서 수행되었다. Alpha11은 모두 128개의 알파 프로세서 시스템을 이더넷과 미리넷으로 연결한 클러스터로 대단위 연산에 필요한 컴퓨팅 자원을 제공하기위해 제작되었다. 우리는 삼성 종합 기술원 CSE 센터 담당자의 도움으로 128개의 노드 중 32개를 할당 받아 사용할 수 있었다.

3. 최적 프로세서 수 효용성 검증

우리는 통신 지연 모형을 이용해 이론적으로 구한 최적

프로세서 수를 실제 실험 결과와 비교하여 최적 프로세서 수의 효용성을 검증하였다. 실험 결과는 수많은 요소에 의해 영향을 받아 오차를 내포한다. 특히 사용하는 프로세서의 수가 많아질수록 연산 시간이 지수적으로 감소하기 때문에 상대적인 오차는 더 커질 수 밖에 없다. 이론적으로 구한 최적 프로세서 수가 실험적으로 구한 것과 비

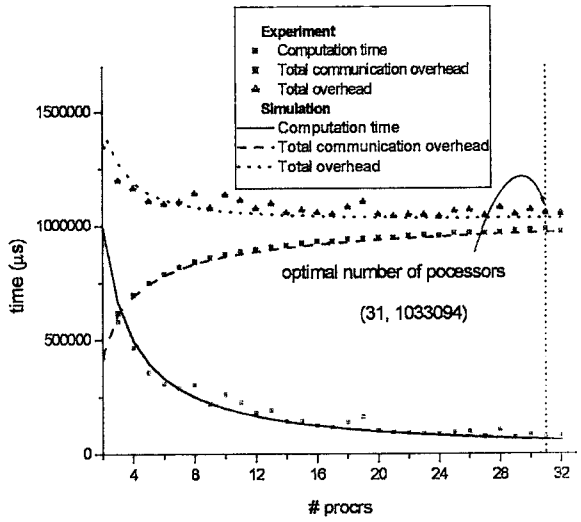

그림 1 병렬 웨이브렛 프로그램의 지연 시간

교하여 허용할 만한 범위 내에 있어야만 최적 프로세서 수가 실질적인 효용성을 가지는 것이다.

우리는 최적 프로세서 수의 효용성을 검증하기 위해 이산 웨이브렛 변환(DWT) 알고리즘을 병렬화하여 실험에 사용하였다. DWT는 입력 신호를 대강의 형태로 분해함으로써 서로 다른 분해능을 가지고 서로 다른 주파수 영

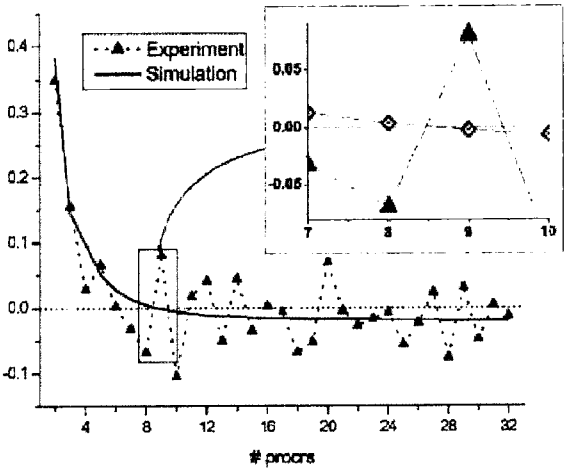

그림 1) 병렬 웨이브렛 프로그램의 최적 프로세서 수

역에서 신호를 분석할 수 있다.

그림 2는 병렬 웨이브렛 프로그램의 사용 프로세수 수에 따른 지연 시간을 나타낸다. 통신 시간이 연산 시간을 상회하는 때는 4개의 프로세서를 사용한 때로 통신 시간이 연산시간보다 크더라도 전체 지연시간은 지속적으로 줄고 있다. 가장 작은 지연 시간을 가지는 때는 31개의 프로세서를 사용한 때이지만 주변과 비교하여 성능의 차이가 미비하고 오차의 영향을 더욱 많이 받는다.

그림 3은 최대 한계 효율비 C를 1%로 했을 때 식 (3)을 프로세서 수에 따라 계산한 결과이다. 실선은 이론적으로 구한 값이고 점선은 실험을 통해 측정한 값이다. 최적 프로세서 수는 식(3)이 0보다 큰 최대 프로세서 수인 8이다. 이때 속도 향상 비율은 1.6801 이며 효율은 0.2801이다. 오차의 표준편차는 4%로 실험값과 이론값이 매우 근사함을 알 수 있다.

3. 결론

본 연구를 통하여 우리는 병렬 시스템의 성능과 비용을 효과적으로 이끌어낼 수 있는 최적 프로세서 수를 결정하는 방법을 제안하였다. 최적 프로세서 수는 또한 구체적인 확장성 기준으로도 사용될 수 있다.

제안된 최적 프로세서 수는 기존의 확장성 기준과 달리 구체적인 프로세서 수를 제공하여 시스템의 종류에 상관없는 객관성을 지니며, 고정 시간 형태의 프로그램뿐 아니라 고정 크기 형태의 프로그램에도 적용할 수 있는 범용성, 그리고 병렬 시스템과 프로그램의 조합의 영향을 동시에 고려하므로 종합성의 특성을 가진다.

신뢰성 있게 최적 프로세서 수를 이론적으로 구하기 위해서는 적절한 통신 지연 모형이 필수적이다. 더욱 합리적이고 논리적인 통신 지연 모형에 관한 연구가 필요하다.

[1] Z. Xu and K. Hwang, " Modeling communication overhead: MPI and MPL performance on the IBM SP" , *IEEE Parallel & Distributed Technology*, Vol. 4, No. 1, Aug, 1996, Page(s): 9--23

[2] Edward A. Luke and Ioana Banicescu and Jin Li, " The Optimal Effectiveness Metric for Parallel Application Analysis" , *Information Processing Letters*, Vol. 66, No. 5, 1998, Page(s): 223—229

[3] X.H. Sun, and L. Ni, " Scalable Problems and Memory-Bounded Speedup" , *Journal of Parallel and Distributed Computing*, Vol. 19, Sept. 1993. Page(s): 27--37

[4] Grama, A., Gupta, A. and Kumar, V. " Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures" , *Technical Report University of Minnesota, IEEE Parallel & Distributed Technology*, Vol.1, No.7, 1993, Page(s): 12—21

[5] Xian-He Sun, Rover, D.T. " Scalability of parallel algorithm-machine combinations" , *Parallel and Distributed Systems, IEEE Transactions*, Vol. 5, 6, June 1994, Page(s): 599—613

[6] R. Hockney, "The Communication Challenge for MPP: Intel Paragon and Meiko CS-2", *Parallel Computing*, Vol. 20, No. 3, 1994, Page(s): 389--398