적응성 있는 부하 재분배를 위한 유전적 방법론

이성훈
천안대학교 정보통신학부
Shlee@cheonan.ac.kr

A Genetic-based Methodology for Adjustable Load Redistribution

Seong hoon Lee
Information&communication Div. Cheonan Univ.

요 약

송신자 개시 부하 균등 알고리즘에서는 전체 시스템이 과부하일 때 송신자(과부하 프로세서)가 부하를 이전하기 위해 수신자(저부하 프로세서)를 발견할 때까지 불필요한 이전 요청 메시지를 계속 보낸다. 따라서 이 같은 상황에서는 저부하 상태인 수신자 프로세서로부터 승인 메시지를 받기까지 불필요한 프로세서간 통신으로 인하여 프로세서의 이용률이 저하되고 또한 태스크의 처리율이 낮아지는 문제점이 발생한다. 본 논문에서는 이질형 분산 시스템에서의 동적 부하 균등을 위해 진화알고리즘을 기반으로 하는 접근 방법을 제안한다.

Abstract

In a sender-initiated load balancing algorithm, a sender continues to send unnecessary request messages for load transfer until a receiver is found while the system load is heavy. Therefore, it yields many problems such as low cpu utilization and system throughput because of inefficient inter-processor communications until the sender receives an accept message from the receiver in this environment. This paper presents an approach based on evolutionary algorithm. The processors to which the requests are sent off are determined by the proposed algorithm to decrease unnecessary request messages.

1. 서 론

분산시스템에서 부하 균등 알고리즘(load balancing algorithm)은 크게 3가지 -정적, 동적, 적응적 방법- 로 분류할 수 있다. 본 논문에서는 이러한 방법들 중 동적 부하 균등 알고리즘을 기반으로 한다. 동적(dynamic) 부하 균등 알고리즘은 실행 중에 과부하(overloaded) 프로세서가 시스템의 부하정보를 이용하여 초과된 태스크를 저부하(underloaded) 프로세서로 보낸다. 기존에 연구되어온 많은 부하 균등 알고리즘들은 대부분 동형 분산 시스템(homogeneous distributed systems)환경에서 부하를 각 프로세서에 균등하게 유지하기 위한 연구들이었다[1, 2, 3, 4]. 이러한 기존의 알고리즘들은 각 프로세서들이 서로 다른 처리 속도를 갖는 이질형 분산 시스템에서는 적합하지 않다.

동적 부하 균등 기법은 3 방법 - 송신자 개시 방법(sender-initiated), 수신자 개시 방법(receiver-initiated), 대칭 개시 방법(symmetrically-initiated) - 으로 세분화할 수 있다. 본 논문에서는 이들 중 송신자 개시 방법을 기반으로 한다. 송신자 개시 방법은 송신자 프로세서(sender : overloaded processor)가 수신자 프로세서(receiver : underloaded processor)에게태스크를 전송하기 위해 부하 균등 알고리즘을 수행한다[1, 2].

분산 시스템내의 전체적인 시스템 부하가 과부하 상태이면 태스크를 이전 받을 수 있는 프로세서를 쉽게 찾을 수 없다. 왜냐하면 대부분의 프로세서들이 송신자 프로세서로부터 태스크를 이전 받을 수 있는 저부하 상태를 유지하지 못하기 때문이다. 따라서 이러한 과정을 승인 메시지를 받을 때까지 반복적으로 임의로 선정된 다른 프로세서

들에 태스크 이전 요청 메시지를 보내야 한다. 그러므로 많은 이전 요청 및 거절 메시지들이 발생하고 이로 인하여 실행 전에 많은 시간이 소모되며 실질적인 태스크 처리 시간이 낭비된다.

2. 부하 균등

전체 분산 시스템의 부하 균등을 위해서 모든 프로세서의 부하 상태를 3단계 (저부하, 정상부하, 과부하)로 분리하여 사용하며 이들은 각 프로세서에서의 VCQL로 결정된다. 이 같은 3단계 부하 상태를 유지하기 위하여 2개의 임계값 즉, 하한 및 상한 임계값을 사용한다.

2.1 부하 척도(load measure)

분산 시스템내 각 프로세서의 부하 상태를 측정하기 위한 척도로 많은 방법들 중에서 CPU 큐 길이를 선정하여 사용하였다. 이 같은 이유는 기존의 연구에서 가장 적합한 것으로 알려져 있기 때문이다[7]. 하지만 이질형 분산 시스템의 가장 큰 특징은 동일한 태스크에 대하여 각기 다른 프로세서들이 다른 처리 시간을 갖는다는 것이다. 따라서 기존의 부하 척도 방법인 CPU 큐 길이는 이질형 분산 시스템의 부하 상태를 적절하게 반영하지 못한다고 할 수 있다. 이러한 배경 하에 이질형 분산 시스템의 부하 상태를 적절히 나타낼 수 있는 다음과 같은 부하 척도 방법 - VCQL(Virtual CPU Queue Length)- 을 사용한다.

$$VCQL = T_{no} \times P_{tu} \qquad (1)$$

매개변수 T_{no}는 CPU 큐에 있는 태스크의 개수를 의미하고 P_{tu}는 각 프로세서에서 하나의 태스크를 처리하는데 소요되는 처리시간을 나타낸다.

2.2 코딩 방법

분산 시스템 내에 있는 전체 프로세서의 개수가 n일 때 i번째 프로세서는 P_i로 표현되며 i는 $0 \le i \le n-1$의 범위를 갖는다. 이러한 분산 시스템 내 각 프로세서는 자신의 집단을 갖으며 각 집단은 여러개의 스트링들로 구성된다. 이러한 스트링을 표현하는 방법으로 본 논문에서는 이진 코딩 방법을 이용한다.

2.3 적합도 함수

한 집단에 포함된 각 스트링은 다음 적합도 함수(fitness function)로 평가된다.

$$F_i = \left(\frac{1}{\alpha \times TMPT + \beta \times TMTT + \gamma \times TTPT} \right) \qquad (2)$$

식 (2)에서 사용된 α, β, γ는 각 매개변수 TMPT, TMTT, TTPT에 대한 가중치로서 4절의 성능 평가에서 다룬다. TMPT는 전송될 요청 메시지들의 처리 시간으로서 전송될 요청 메시지의 개수(Request Message Number)와 각 메시지 처리 시간의 곱(product)으로 정의할 수 있다.

$$TMPT = \sum_{k \in x} (ReMN_k \times \text{Time Unit}),$$
$$x = \{ i \mid v_i = 1 \quad \text{for} \quad 0 \le i \le n-1 \} \qquad (3)$$

매개 변수 TMTT는 송신자로부터 선정된 스트링내 '1'로 설정된 각 비트에 대응되는 수신자 대상 프로세서들까지의 메시지 전송 시간의 합(summation)을 의미한다. 여기서 n은 분산 시스템 내 전체 프로세서의 개수가 된다.

$$TMTT = \sum_{k \in x} (EMTT_k),$$
$$x = \{ i \mid v_i = 1 \quad \text{for} \quad 0 \le i \le n-1 \} \qquad (4)$$

식 (4)에서 사용된 매개 변수 $EMTT_k$(Each Message Transfer Time)는 송신자 프로세서로부터 k번째 프로세서까지의 메시지 혹은 태스크 전송 시간을 나타낸다. 마지막으로 TTPT는 선정된 스트링내 '1'로 설정된 각 비트에 대응되는 수신자 대상 프로세서들에서 태스크들을 처리하는데 소요되는 시간으로서 식 (5)와 같이 정의할 수 있다.

$$TTPT = \sum_{k \in x} (VCQL_k),$$
$$x = \{ i \mid v_i = 1 \quad \text{for} \quad 0 \le i \le n-1 \} \qquad (5)$$

2.4 알고리즘

본 논문에서 제안하는 알고리즘은 5개의 프로시쥬어로 구성되는데 이 같은 프로시쥬어들은 분산시스템내 각 프로세서에서 시행된다. 알고리즘을 구성하는 각 프로시쥬어들의 기능은 다음과 같다.

○ 초기화(initialization): 초기화는 전체 시스템이 작업을 시작할 때만이 각 프로세서에서 시행된다. 스트링들로 구성된 집단은 어떠한 스트링의 중복 없이 임의로 만들어진다.

○ 부하 측정(load_check): 어떤 하나의 프로세서에서 새로운 태스크가 들어올 때마다 부하 측정 프로시쥬어가 호출되어서 자신의 부하를 측정한다.

○ 스트링 평가(string_evaluation): 스트링 평가 프로시쥬어는 집단 내에 존재하는 스트링들의 적합도를 계산한다.

○ 진화 연산(evolutionary_operation): 돌연변이 및 선택, 교배 등을 포함하는 진화 연산 프로시쥬어는 다음과 같은 방법으로 해당 집단에 적용된다. 진화 연산은 해당 스트링의 적합도에 비례하는 확률로서 집단내에 존재하는 스트링들중 하나의 스트링을 선정한다.

3. 성능 평가

3.1 실험 환경

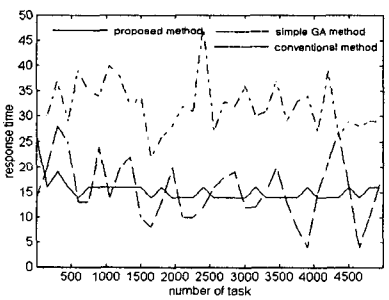
본 논문에서의 시스템은 이질형 시스템(heterogeneous system)으로서 30개의 이질형 프로세서로 구성되었다고 가정한다. 또한 처리될 각각의 태스크 크기는 동일한 것으로 가정한다. 각각의 프로세서로부터 다른 프로세서로의 메시지 전송 및 태스크 전송 시간은 2차원 배열로 표현되며 이때 각 원소의 값은 난수 발생기(random number generator)로 부터 결정되는 3 이하의 값을 갖는 것으로 하였다. 표 1은 실험에 사용된 매개 변수 내용이다.

<표 1> 매개변수들의 내용

프로세서의 개수	30
교배 발생 확률(P_c)	0.7
돌연변이 발생 확률(P_m)	0.1
스트링(염색체)의 개수	50
처리될 태스크 개수	5000
스트링의 부분 개수(p)	5
TMPT에 대한 가중치(α)	0.5
TMTT에 대한 가중치(β)	0.15
TTPT에 대한 가중치(γ)	0.05

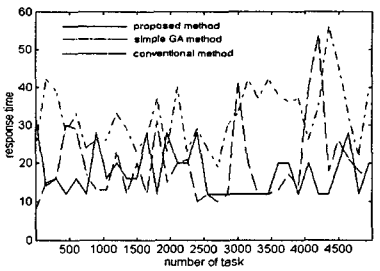
4.2 비교 평가

[실험 1] 본 실험은 분산 시스템의 전체 시스템 부하가 60%일 때 제안된 방법과 기존의 송신자 개시 방법 및 단순 진화 알고리즘을 이용한 접근법간의 반응 시간을 알아보기 위한 실험이다.



[그림 1] 실험 1에 대한 결과

[실험 2] 본 실험은 전체 시스템 부하가 80%일 때의 반응 시간을 알아보기 위한 것이다.



[그림 2] 80%일 때 반응 시간

5. 결론

본 논문에서는 이질형 분산 시스템에서의 효율적인 부하 균등의 문제점을 해결하기 위하여 진화 알고리즘을 기반으로 하는 새로운 동적 부하 균등 기법을 제안하였다. 이러한 진화 알고리즘을 통하여 이전 요청 메시지들이 전송될 수신자 대상 프로세서들을 결정한다. 본 논문에서는 이질형 분산 시스템의 특성 중에서 각 프로세서의 처리 시간이 다르다는 점만을 고려하였다.

[참고 문헌]

[1] D.L. Eager, E.D. Lazowska, J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Transactions on Software Engineering*, Vol.12, No.5, pp.662-675, May 1986.

[2] N.G. Shivaratri, P. Krueger and M. Singhal, "Load Distributing for Locally Distributed Systems," *IEEE Computer*, Vol.25, No.12, pp.33-44, December 1992.

[3] L.M. Ni, C.W. Xu and T.B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing," *IEEE Transactions on Software Engineering*, Vol.SE-11, No.10, pp. 1153-1161, October 1985.

[4] M. Livny and M. Melman, "Load Balancing in Homogeneous Broadcast Distributed Systems," *Proc. ACM Computer Network Performance Symp*, pp.44-55, 1982.

[5] Garrison W. Greenwood, Christian Lang and Steve Hurley, "Scheduling Tasks in Real-Time systems Using Evolutionary Strategies," *Proc. Third Workshop on Parallel and Distributed Real-Time Systems*, pp.195-196, 1995.

[6] David B. Fogel and Lawence J. Fogel, "Using Evolutionary Programming to Schedule Tasks on a Suite of Heterogeneous Computers," *Computers & Operations Research*, Vol. 23, No.6, pp.527-534, 1996.

[7] T. Kunz, "The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme," *IEEE Transactions on Software Engineering*, Vol.17, No.7, pp. 725-730, July 1991.