부하 분산을 위한 지능형 예약 알고리즘

이준연*

An Intelligent Reservation Algorithm for Workload Distribution

Jun-Yeon Lee*

요 약

본 논문에서는 클라이언트들의 작업부하를 분산하기 위한 측정 알고리즘을 제안하였다. 이 알고리즘은 작업들이 재배치되는 것을 허용할 때, 송신자가 수신자에게 요청한 처리량만큼 적절하게 전송하여야 한다. 이 양은 송신자와 수신자의 협상 과정에서 동적으로 결정된다.

전송하여야 할 작업량을 결정할 때 전송 노드는 수신하는 다른 노드들의 처리 속도를 포함하여 송신 노드와 수신 노드의 현재 부하상태, 재배치하기에 적절한 작업의 처리 요구량 등의 요소들을 고려하여야 한다.

이러한 분석에 근거하여 이기종 분산 웹서버 시스템에 적합한 새로운 알고리즘을 제안하고, 기존 알고리즘과의 성능을 비교하였다. 시뮬레이션을 통하여 본 논문에서 제안한 알고리즘이 CPU 이용율과 응답시간에서 성능의 개선을 보였다.

ABSTRACT

We proposes an algorithm of measuring computation loads to distribute workload of clients. The key of the algorithm is to transfer a suitable amount of processing demand from senders to receivers in relocating intrinsically. This amount is determined dynamically during sender-receiver negotiations.

Factors considered when this amount is determined include processing speeds of different nodes, the current load state of both sender and receiver, and the processing demands of tasks eligible for relocation. We also propose a load state measurement scheme which is designed particularly for heterogeneous systems.

Based on this analysis, we design new algorithm for supporting heterogeneous distributed web server system, and compare their performance against other existing algorithms. We show that the new algorithm improve the performance of CPU utilization and response time.

키워드

예약 프로토콜, 부하 분산, 연산부하 측정

Ⅰ. 서 론

웹의 폭발적인 성장과 더불어 인기있는 웹 사이트는 많은 양의 인터넷 트래픽을 처리하게 된다. 이러한 사이트들은 클라이언트들에게 적은 비용으로 더 나은 서비스를 제공하기 위하여 서비스 요청을 분산시켜야 한다. 따라서 현재 시스템의 부하 정보를 사용하여 응용 태스크들을 분산 웹 서버의 각 노드에 재배치함으로써 시스템의 성능과 작업의 응답시간을 개선할 수 있다 [1]. 대부분의 기존 부하 분산 알고리즘은 동종 시스템

* 동명정보대학교

접수일자 : 2005. 3. 22

을 대상으로 하고 있다[2, 7, 9]. 이러한 시스템은 다른 처리 성능을 가진 이기종 시스템에서는 적합하지 않다.

본 논문에서는 작업의 처리시 요구사항과 종류가 다른 작업들을 위한 이기종 분산 웹서버 시스템에서 예약 기법을 이용한 동적 부하 균등화 알고리즘을 제안한다. 예약 기법은 3가지 주요 요소로 구성되어 있다. 첫번째는 많은 수의 작업들이 원격 할당에 선택될 수 있도록 허용하는 배치(batch) 작업 할당 부분이다[3, 4]. 둘째는 예약 프로토콜로, 이것은 작업 배치에 포함된 부하들의 양을 송신자와 수신자가 협상하는 것이다. 그리고 마지막으로 대칭형 초기화 위치 정책으로, 시스템 부하의 변화가 생겼을 때, 이를 감지하고 대응하는 적응력있는 알고리즘이다. 지능형 예약 알고리즘의 주요 설계 정책은 작업 배치(batch) 구성으로, 이것은 작업 배치를 구성하는 후보들 중에서 일련의 작업을 선택하는 것이다. 시뮬레이션을 통하여 예약 기법이 서로 다른 이기종 시스템에 더 적합하고, 더 나은 성능 개선을 보여줄 것이다.

이 논문은 2장에서는 예약 알고리즘을 설명하고, 3장에서는 시뮬레이션에 의한 예약 프로토콜을 평가하고, 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 예약 알고리즘

지역 작업이 도착하면, 작업 할당 후보가 되기 위하여 할당 큐로 진입한다. 이 작업은 지역 임계큐에 진입함으로써 지역 작업이 되거나, 혹은 원격 노드의 임계큐에 진입함으로써 원격 작업이 된다. 일단 작업이 임계큐에 진입하면 다른 노드로 재할당될 수 없다. 즉, 작업 재할당을 허용하지 않는다. 임계큐에 있는 작업은 서비스큐의 용량이 최대값에 도착하지 않았을 때만 서비스 큐로 옮겨질 수 있다. 서비스큐에 있는 작업은 CPU에 의해서 라운드로빈으로 처리된다. CPU에 할당된 작업은 매번 정해진 시간 단위인 TCPU만 사용한다. 이 시간동안 작업이 종료되지 않으면 서비스큐의 맨 뒤로 되돌아간다.

예약 기법의 주요 핵심은 일련의 작업을 할당하는 정책으로, 이는 송신자와 수신자의 협상 과정에서 여러 작업이 원격 전송을 위하여 선택될 수 있다는 것이다. 이 때, 처리 능력이 서로 다르기 때문에 송신 노드로부터 전송되는 작업이 수신 노드에게 거의 영향을 미치지 않거나 혹은 매우 심각한 과부하 현상이 발생할 수 있다. 이것은 일반적으로 사용되는 단일 작업 전송 방법이 이기종 시스템에는 적합하지 않다는 것을 의미한다.

2.1 위치 정책

위치 정책이란 적절한 작업 전송 대상을 찾는 정책을 말한다. 가장 일반적으로 사용되는 정책은 폴링인데, 이는 과부하 노드가 자신의 부하를 전송하기 위한 저부하 노드를 찾기 위하여 시작할 수도 있고, 혹은 저부하 노드가 자신에게 부하를 전송하고자 하는 과부하 노드를 찾을 경우에도 시작할 수 있다. 전자를 송신 노드 시작 정책이라 하고, 후자를 수신 노드 시작 정책이라 하며 이 두가지가 모두 사용되는 경우를 대칭형 위치 정책이라 한다.[7]

본 논문에서는 Shivaratri와 Krueger가 제안한 대칭형 위치정책을 사용하는데, 이 방법은 시스템의 부하가 낮을 경우에는 송신노드 시작 정책을 사용하고, 시스템의 부하가 높을 경우에는 수신노드 시작 정책을 사용한다. 이 방법은 시스템 부하의 전체 영역에서 강점을 갖는다. 따라서 예약 기법은 대칭형 위치정책이며, 폴링 메시지의 수를 제어하기 위하여 폴링 한계(PL:polling limit)를 사용한다[1,3,11].

2.2 예약 프로토콜

예약 프로토콜은 프로세서 쓰레싱을 해결하기 위하여 고안된 것으로 기본적인 방법은 다음과 같다. 송신 노드와 수신 노드의 쌍이 결정되면 송신 노드는 폴링 메시지에서 수신 노드로의 전송을 보장할 수 있는 작업의 양을 정의한다. 이 값을 근거로 수신 노드는 송신 노드로부터 받아들일 작업의 양을 결정한다. 예약값은 송신 노드로부터 약속된 양만큼 받기 위한 할당량으로서, 그리고 다른 송신 노드로부터의 작업 송신을 막기 위하여 사용된다[5].

바꾸어 말하면, 수신 노드는 예약값만큼 자신의 부하를 증가시킴으로써 다른 잠정적인 송신노드와의 협상 요청을 막고, 이미 작업의 전송이 시작된 것처럼 보장하기 위하여 예약값을 사용한다. 따라서 수신 노드는 다른 송신 노드가 작업 전송을 요청할 경우, 이미 송신 노드가 작업군을 전송중인 것처럼 동작한다. 작

업군의 전송이 완료되면, 이에 상응하는 할당량을 증가시킨다.

대칭형 위치정책을 사용하기 때문에 수신 노드도 폴링을 시작할 수 있다. 수신 노드로부터 송신 노드로 전송된 폴링 메시지는 예약값을 운반한다. 과부하 노드의 작업을 가능한 빨리 분산시키고, 저부하 노드의 역량을 충분히 사용하기 위하여 다른 송수신 노드와 협상중인 노드에서 여러 개의 병행 세션을 허용한다[11].

2.3 작업 부하 측정

작업의 종류가 하나만 존재한다면, 한 노드의 부하 상태는 그 노드에 존재하는 작업의 수로서 표현될 수 있다. 그러나 작업의 종류가 여러 가지일 경우에는 어떤 작업이 다른 작업에 비해 더 많은 서비스 요구량을 필요로 할 것이다. 그리고 동일한 서비스 요구량이 다른 종류의 노드에서는 다른 CPU 시간을 필요로 한다. 이러한 사실로 미루어 일반적으로 작업의 수가 동종 시스템에서조차 부하의 양을 적절하게 나타내지 못한다.

따라서 본 논문에서는 임의의 노드 P_i의 작업 부하 W_i를 P_i에 존재하는 모든 작업의 서비스 요구량의 합을 P_i의 처리 효율로 나눈 것으로 가정한다.

그리고, 노드 P_i의 작업 부하를 근거로 가상 부하 VW_i를 다음과 같이 정의할 수 있다.

$$VW_i = W_i^{JQ} + REG_i - GUR_i$$

여기서 REG_i는 자신의 노드로 이주되기로 예약된 부하의 크기이며 GUR_i는 다른 원격 노드로 이주를 보장받은 값이다.

한 노드의 부하 상태를 나타내는 경우 3가지 상태로 충분하다[1,6]. 예약 알고리즘에서 그들의 가상 부하에 따라 3가지 상태 중 하나를 가지게 된다. 이 상태는 표 1과 같다.

표 1. 3단계 작업측정
Table 1. The 3-level workload measuring

부하 상태	기준
underload	$VW_i \le T_{low}$
normalload	$T_{low} < VW_i \le T_{up}$
overload	$VW_i > T_{up}$

T_{up}과 T_{low}는 알고리즘을 설계할 때 사용되는 매개변수이며, 이는 각각 상한 임계값과 하한 임계값이다.

2.4 전송량 결정

송신 노드에서의 보장값 *gur*은 예약 알고리즘의 성능에 지대한 영향을 미친다. *gur* 값을 너무 크게 설정하면 다른 송신 노드들이 현재 협상중인 노드를 제외한 다른 잠정적인 수신 노드들에게 작업을 전송할 수 있는 기회를 빼앗게 된다. 그 결과 분산될 수 있는 작업의 크기가 제한된다. 반대로 *gur*의 크기를 너무 작게 설정하면 예약 알고리즘의 장점을 충분히 살릴 수 없게 된다.

*gur*의 크기는 다음과 같이 정해질 수 있다.

$$gur^{(s)} = min \left\{ \begin{array}{c} (T_{up} - T_{low}) * p_r^s \\ W_s^{AQ} \end{array} \right.$$

여기서 위첨자 $^{(s)}$는 송신 노드 P_s를 의미한다. 예를 들어 $gur^{(s)}$=2.5라면 이는 송신 노드 P_s에서 실행될 때 2.5 단위 시간이 소요되는 작업을 나타낸다.

이 식을 증명하기 위하여 다음과 같은 규칙을 정한다.

가정 1: 전체 서비스 요구량이 $d^{(r)}$인 작업이 도착하더라도 수신 노드 P_r이 overload 상태가 되지 않는다. 즉,

$$\neg (VW_r + d^{(r)} \Rightarrow \text{overoad})$$

앞의 표 1.에서 정의한 부하 측정 기준에 의하여 가정 1은 다음과 같이 표현될 수 있다

$$VW_r + d^{(r)} \le T_{up} \quad (1)$$

이 식(1)은 다시 아래와 같이 표현될 수 있다.

$$d^{(r)} \le (T_{up} - VW_r) \quad (2)$$

표 1.에 의하여 수신 노드의 가상 부하는 0과 *lower_threshold*사이의 값이 된다. 그러므로 VW_r의 상한값과 하한값은 다음과 같다.

Minimum: $VW_r = 0$
Maximum: $VW_r = T_{low}$

위의 식 (2)는 다음과 같이 다시 쓰여질 수 있다.

$$d^{(r)} \leq \begin{cases} T_{up} & \text{if } VW_r = 0 \\ T_{up} - T_{low} & \text{if } VW_r = T_{low} \end{cases} \quad (3)$$

식 (3)의 두 번째 경우에서는 송신 노드 P_s가 기대할 수 있는 수신 노드 P_r의 최대 수신 작업 부하의 양은 $(T_{up} - T_{low})$이다. 따라서 이 양을 송신 노드 P_s에서 실행될 때 필요한 CPU 시간으로 나타내면 $(T_{up} - T_{low}) \cdot p_r^s$이다.

그러므로 $gur^{(s)}$는 이 값보다 더 큰 값을 가질 수 없다. 이로써 식 (1)이 증명되었다. 식 (1)의 두 번째는 송신 노드 P_s가 자신의 작업 큐에 있는 작업량 이상을 보장값으로 사용할 수 없다는 것이다.

$gur^{(s)}$이 도착하면 수신 노드 P_r은 이에 대응하는 예약값 res를 계산하여야 한다. 가정 4)과 자신의 로컬 부하량을 기초로 하여 식 (2)가 참이 되는지를 결정하여야 한다. 자신이 수신할 수 있는 최대값은 다음과 같다.

$$d^{(r)} = T_{up} - VW_r \quad (4)$$

수신 노드 P_r는 자신의 처리 용량보다 송신 노드가 전송한 $gur^{(s)}$의 값이 더 큰 경우 이를 예약값으로 설정할 수 없다. 따라서 수신 노드의 예약값 $res^{(r)}$은 다음과 같이 결정된다.

$$res^{(r)} = \min \begin{cases} d^{(r)} = T_{up} - VW_r \\ gur^{(s)} \cdot p_s^r \end{cases} \quad (5)$$

그러나 식(1)에서 $gur^{(s)}$의 최대값은 $[(T_{up} - T_{low}) \cdot p_r^s]$이거나, 혹은 P_r의 CPU 시간이 계산된 경우에는 $(T_{up} - T_{low})$이다. 반면에 수신 노드 P_r의 VW_r의 최대값은 T_{low}이므로 $d^{(r)}$의 최소값은 $(T_{up} - T_{low})$이다. 따라서 $gur^{(s)} \cdot p_r^s$는 최소한 $d^{(r)}$의 크기가 되므로 식 (5)는 $res^{(r)} = gur^{(s)} \cdot p_r^s$와 같이 간단하게 사용할 수 있다.

수신 노드가 폴링을 시작하는 경우에는 공식 (4)는 사용할 수 없다. 따라서 보장값은 다음과 같이 정의할 수 있다.

송신 노드 시작의 경우:

$$res^{(r)} = gur^{(s)} \cdot p_r^s$$

수신 노드 시작의 경우:

$$res^{(r)} = T_{up} - VW_r$$

전송할 작업량의 크기를 조절하는 다음 단계는 송신 노드가 $res^{(r)}$의 범위 내에서 가장 큰 작업 서비스 요구량을 결정하는 것이다. 이 양을 Θ라고 표시하면, Θ를 결정하기 위하여 다음의 두가지 규칙을 정한다.

가정 2 : 총 CPU 시간이 $\beta^{(s)}$인 작업을 제거하더라도 송신 노드의 상태가 underload상태가 되지 않는다. 즉,

$$\neg (VW_s - \beta^{(s)} \Rightarrow \text{underload})$$

표 1.에 의하여 $(VW_s - \beta^{(s)} > T_{low}$가 된다. 즉,

$$\beta^{(s)} < VW_s - T_{low} \quad (7)$$

송신 노드는 수신 노드가 예약한 부하보다 더 큰 작업부하를 전송할 수 없으므로

$$\Theta = p_{T(Ps)} \cdot \min \begin{cases} \beta^{(s)} \\ res^{(s)} \cdot p_r^s \end{cases} \quad (8)$$

여기서 Θ는 서비스 요구 단위로 측정된다. 공식 (6)에 의하여 $res^{(r)}$의 최대값은 T_{up}이다. 따라서 공식 (10)에서 Θ의 최대값은 $(T_{up} \cdot p_{T(Pr)})$이 된다. 바꾸어 얘기하면, $(T_{up} \cdot p_{T(Pr)})$은 송신 노드가 전송할 수 있는 작업 전송량의 최대 크기가 된다. 수신 노드의 처리 효율 p가 커질수록 전송할 수 있는 작업량의 크기는 더 커진다.

Θ의 크기가 결정되면 송신 노드는 전송할 작업군에 포함될 작업을 선택하여야 한다. 이 때 수행되는 정책을 작업 구성(TC:Task Composition) 정책이라 한다.

작업 구성 정책은 또다른 연구분야이므로 본 논문

에서는 언급하지 않고, Jerrell이 제안한 "Second-order implicit diffusion algorithm"에 따르고자 한다.[5]

Ⅲ. 성능 평가

본 논문에서는 폴링 메시지를 처리하기 위한 CPU 오베헤드는 무시할 수 없으므로 $CPU_{polling}$으로 표시한다. 폴링 메시지는 속성상 길이가 짧기 때문에 하나의 메시지를 작성하는데 소요되는 비용을 $F_{polling}$이라 하고, D를 네트워크 전파 지연시간이라 하면, 하나의 폴링 메시지를 전송할 때의 지연시간은

$$DELAY_{polling} = F_{polling} + D$$

와 같이 정의한다.

송신 노드와 수신 노드에서 송수신하는 작업은 동일한 양의 CPU 오버헤드를 가지는데, 다음과 같이 정의된다.

$$CPU_{task} = C_{LB} + C_{pack} * \sum_{i=1}^{b} l_{c(i)}$$

여기서 C_{LB}는 부하균등화 알고리즘을 실행하는데 소요되는 CPU 시간을 나타내는 상수이며, C_{pack}은 각 작업 메시지를 작성하고 해제하는데 소요되는 시간 상수이다.

작업을 전송하는데 소요되는 통신 지연시간은 다음과 같이 정의한다.

$$DELAY_{task} = (F_{task} * \sum_{i=1}^{b} l_{C(i)}) + D$$

여기서 F_{task}는 통신채널로 하나의 작업을 전송하는데 소요되는 시간이다.

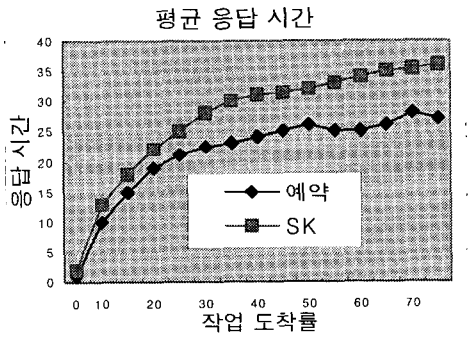
시뮬레이션에서 사용된 매개변수의 값은 표 2에서 보는 바와 같다. 적응 알고리즘은 Shivaratri와 Krueger의 대칭형 위치정책(이하 SK 알고리즘이라 칭함)과 단일 작업 할당을 사용하는데, 이 정책이 비교 대상이다[10]. 예약 알고리즘과 SK 알고리즘의 위치정책은 동일한 메커니즘을 사용하였다.

표 2 시뮬레이션 매개변수 값

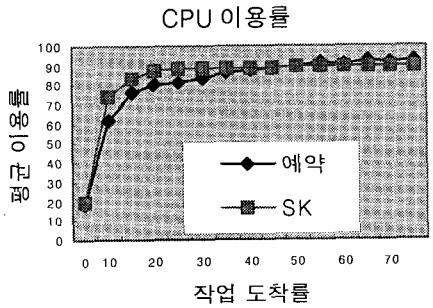
매개변수	값	매개변수	값
Q_i, $\forall i \in M$	30	PL	5
T_{CPU}	0.1	l_i	5
T_{low}	5	$CPU_{polling}$	0.005
T_{up}	20	$F_{polling}$	0.001
F_{task}	0.001	C_{pack}	0.001
D	0.010	C_{LB}	0.002

그림 1의 (a) 그래프는 10,000개 이상의 작업에 대하여 작업의 평균 응답 시간을 보여준다. 예약 알고리즘은 나 알고리즘에 비해 더 적은 응답시간을 보여주고 있다.

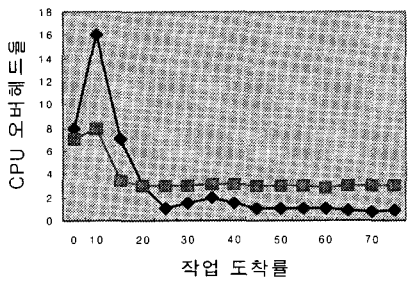
(b) 그래프는 작업 도착률이 35% 이하일 때 더 낮은 CPU 이용률을 보여주고 있다. 이것은 SK알고리즘과 비교할 때 빠른 처리노드에서는 처리시간이 더 적음을 시사한다.



(a)

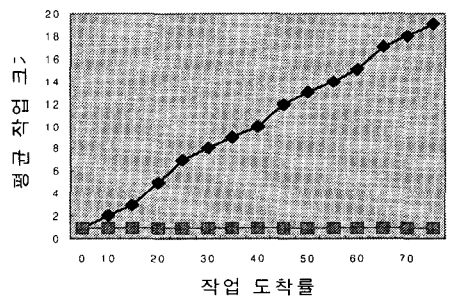


(b)



(c)

(c)에서는 작업도착률이 20% 이하일 때 더 나은 성능을 보이나, 시스템의 부하가 높아지면 반대 현상이 발생한다. 예약 알고리즘이 저부하 상태에서는 CPU 오버헤드가 약 50~70%가량 줄어들고, 중간부하에서 높은 부하 영역에서는 더 효율적인 협상 알고리즘을 사용할 수 있다. 만약 전송할 작업량이 b개라고 가정하면 동일한 작업량을 전송할 경우 b번의 성공적인 협상이 필요하므로 예약 알고리즘이 더 효율적이다.



(d)

Ⅳ. 결 론

본 논문에서는 이기종 분산 시스템에 적합한 예약 알고리즘을 제안하였다. 이 알고리즘의 중요한 요소는 송신노드에서 수신노드로 적합한 양의 작업 부하를 전송하는 것이다. 이 전송량은 예약 프로토콜에 의하여 동적으로 결정된다. 그리고, 또한 작업부하 측정 정책을 사용하여 가상부하를 제안하였다.

알고리즘의 성능은 시뮬레이션을 통하여 평가되었다. 결과를 통하여 작업 응답시간과 CPU, 그리고 통신 오버헤드가 개선되었음을 보여주었다.

참고문헌

[1] Agarwal, P., MTech Thesis, CSE, IIT Kanpur. "A testbed for performance evaluation of load balancing strategies of web server system", May, 2001.

[2] Cisco Systems Inc. "Distributed Director White Paper". http://www.cisco.com/warp/public/cc/cisco/mkt/scale/distr/tech/d_wp.htm

[3] "Apache Keep-Alive support." This document can be obtained from http://www.apache.org/docs-1.2/keepalive.html.

[4] Daemon9. Libnet: Network Routing Library, Aug. 1999 This document can be obtained from http://www.packetfactory.net/libnet/.

[5] Mosberger, D., and Jin, T. httperf-A Tool for Measuring Web Server Performance. This document can be obtained from ftp://ftp.hpl.hp.com/pub/httperf/.

[6] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on Web-server systems. IEEE Internet Computing, 3(3):pp.28-39, May 1999.

[7] M. F. Arlitt, R. Friendrich, and T. Jin. Workload characterization of a Web proxy in a cable modem environment. ACM Performance Evaluation Review, 27(2):pp.25-36, Aug. 1999

[8] J. E. Pitkow. Summary of WWW characterizations. World Wide Web, 2(1-2):pp.3-13, Mar. 1999

[9] N. G. Shibaratri, and P. Krueger. Two adaptive location policies for global scheduling algorithms. In Proceedings, The 10th International Conference on Distributed Computing Systems, pp. 502-509, May 1990.

[10] T. Schroeder, S. Goddard, and B. Ramamurthy. Scalable Web server clustering technologies. IEEE Network, pp.38-45, May 2000.

[11] 이준연, 임재현, "작업 이주시 보장/예약 기법을 이용한 프로세서 쓰레싱 빈도 감소", 정보처리학회논문지A, 8(2):pp.133-146, Jun. 2001

저자소개

이준연(Jun-Yeon Lee)

1990 중앙대 전자계산학과(공학사)
1992 중앙대 전자계산학과(공학석사)
2000 중앙대 컴퓨터공학과(공학박사)
1992-1994 Microsoft

2000- 동명정보대학교 멀티미디어공학과 조교수
※관심분야 : 홈네트워크, 상황인식, u-Healthcare