클러스터 시스템에서 차별화된 서비스 제공을 위한 메커니즘의 가용도 분석

최창열[0] 김성수
아주대학교 정보통신전문대학원 정보통신공학과
{clchoi[0], sskim}@ajou.ac.kr

Availability Analysis of Differentiated Service Mechanism in Cluster System

Changyoel Choi[0] Sungsoo Kim
Graduate School of Information and Communication, Ajou University

요 약

다양한 인터넷 응용분야에서 사용자와 서비스 제공자간에 서비스 수준 계약(SLA, Service Level Agreement)을 체결하여 차별화된 서비스를 제공하고자 하는데, 계약 조건의 대표적인 예는 일정시간 내에 서비스를 제공받기 위한 서비스 지연시간과 같은 시스템 성능 척도이다. 따라서 본 논문에서는 클러스터 시스템 환경에서 차별화된 서비스 제공을 위한 알고리즘을 제안하고 이에 대한 가용도 분석을 통해 성능 척도와 가용도 사이의 상관관계를 밝히며, 성능 향상을 위한 고려사항을 제시한다.

1. 서비스 클래스별 클러스터 시스템

계획 없이 구성된 클러스터 시스템은 목표로 했던 시스템의 성능이나 가용도 요구 사항을 만족시키기 어렵다. 즉, 서버 수용 능력을 잘못 예측하여 적절하지 못한 서버수로 구성된 경우 갑자기 증가한 사용자들을 처리하지 못하여 성능 측면의 서비스 수준 계약 준수에 실패하게 된다. 또한 결함허용 기법을 갖추지 못한 클러스터 시스템은 예측하지 못한 결함 발생으로 인하여 가용도 요구 사항을 만족시키지 못할 뿐만 아니라 한 서버에서 제공하던 성능 손실로 인하여 전체 성능의 저하를 야기한다. 주로, 예측하지 못한 결함은 하드웨어적인 결함보다는 소프트웨어의 장시간 작동에 의해 발생하는 소프트웨어적인 결함에 의해서 발생한다[1]. 따라서 본 논문에서는 클러스터 시스템 환경에서 차별화된 서비스 제공을 위한 알고리즘을 제안하고 이에 대한 가용도 분석을 통해 성능 척도와 가용도 사이의 상관관계를 밝히며, 성능 향상을 위한 고려사항을 제시한다.

또한 목표로 하는 클러스터 시스템 구성은 7-layer 라우팅 스위치(Routing Switch)를 통해 부하 분배를 수행하고 제공하고자 하는 서비스를 정적, 동적으로 나누어, 각 서비스마다 클러스터 시스템을 구성한다. 따라서 전체 시스템은 정적 서비스를 처리하기 위한 클러스터(S_1, S_2, ... , S_n)와 동적 서비스를 처리하기 위한 클러스터(D_1, D_2, ... , D_n)로 클래스를 구분한다. 이때 클러스터를 물리적으로 나누어 구성하면 서버 재구성의 유연성에 제약이 있으므로 논리적으로 클래스별 클러스터를 분리하는데, 이것은 가상 지역망(Virtual Local Area Netwrok)을 사용하여 vLan1, vLan2 가상 지역망에 의해 연결된다. 또한 각 클러스터 시스템은 가용도 향상을 위해 소프트웨어 재활 메커니즘을 장착한다.

이 논문은 2003년도 두뇌한국21사업에 의하여 지원되었음.

2. 가용도 분석

클러스터 시스템은 동시에 다수의 서버 가동으로 인한 가용도 저하 문제가 파생하는데, 이를 해결하기 위해 각 서버들은 소프트웨어 재활 메커니즘을 장착한다. 한편 대부분 소결합 클러스터 시스템(Loosely-coupled Cluster System)은 어느 한 서버가 동작을 멈추더라도 동일 클러스터내의 다른 서버에는 영향을 미치지 않는다. 또한 대부분의 가용도 분석을 수행한 기존 연구 논문[2,3]에서는 서버의 고장 및 수리 시간을 지수 분포로 가정하고 있다. 그리고 차별화된 서비스 제공을 위한 메커니즘에서 사용하기 위한 가용도 척도를 확보하는 것을 목표로 삼는데 서비스 클래스별 요구사항이 다르므로 양 클러스터를 동시에 분석을 하는 것은 제외하고 각 클러스터 시스템에 대한 가용도를 분석하기 위한 모델링을 수행한다. 물론 분석 모델은 정적 서비스나 동적 서비스를 제공하기 위한 시스템 모두에 적용 가능토록 설계한다. 이상에서 언급한 내용으로부터 차별화된 서비스 제공을 위한 2 계층 클러스터 시스템의 가용도 분석 모델의 기본적인 가정들은 아래와 같다.

- 각 클러스터 시스템은 n 대의 주서버로 연결되어 있으며, 각 서버는 서로 독립적으로 작업을 수행하며, k 대의 여분서버가 존재하여 임의의 주서버에 고장이 날 경우 이의 역할을 대체하며 고장난 서버는 수리된다.
- 주서버와 여분서버의 고장률(λ), 수리률(μ)은 상수이며 고장 및 수리 시간 간격은 지수분포를 따른다.

- 소프트웨어 재활 주기(λ_r)와 소프트웨어 노화현상으로 인해 서버의 상태가 불안정해지는 시간(λ_u)은 지수 분포를 따른다.
- 주서버에서 여분서버로 작업 전이에 필요한 시간은 일정하며, 작업 전이 시간은 평균값이 $1/\mu_s$인 r-stage Erlangian 분포를 따른다.
- 서버 상태가 불안정하여 재활을 수행해야할 경우 소프트웨어 재활에 필요한 시간은 일정하며, 재활 시간은 평균값이 $1/\mu_r$인 r-stage Erlangian 분포를 따른다.

각 클러스터 시스템의 여분서버는 다른 클래스 서비스를 제공하기 위한 서버들이므로 여분서버를 갖추기 위한 추가적인 설치에 의한 비용을 줄일 수 있다. 그리고 여분서버로의 작업 전이 시간과 재활에 필요한 시간을 r-stage Erlangian 분포로 가정한 것은 가용도 분석 모델링을 모든 상태가 memoryless 성질을 갖는 Markov 모델보다는 보다 현실에 근접한 가용도 계산을 가능케 하기 위해 semi-Markov 모델을 선택하기 위함이다.

정상 상태에서 가동되고 있는 서버는 (주서버수, 여분서버수)의 쌍으로 가진 상태 변수{(n,k), (n,k-1), ... , (n,0), (n-1,0), ... , (1,0)}로 표현된다. 장시간 가동 후 소프트웨어 노화현상이 발생하여 클러스터 시스템은 정상 상태에서 불안정 상태{$U_{n,k}$, $U_{n,k-1}$, ... , $U_{n,0}$, $U_{n-1,0}$, ... , $U_{1,0}$}로 전이하며, 불안정 상태에 있는 서버는 결함이 발생하여 작업 전이 상태{S_k, S_{k-1}, ... , S_1}로 들어가거나 소프트웨어 재활을 수행한다{$R_{n,k}$, $R_{n,k-1}$, ... , $R_{n,0}$, $R_{n-1,0}$, ... , $R_{1,0}$}. 불안정 상태에서 주서버에 결함이 발생하면 (with rate $n\lambda_f$) 작업 전이 과정을 통해 k 개의 여분서버가 사용가능할 때까지 시스템의 가동 서버는 n 개로 유지되고 여분서버에 고장이 발생할 경우(with rate $k\lambda_f$) 작업 전이 시간이 불필요하므로 지연시간 없이 상태가 변한다. 주서버 중 $\lceil n/2 \rceil$ 수만 먼저 재활을 수행하는데 나머지 서버들이 수행하던 작업을 인계받은 후 수행하기 때문에 재활 가동으로 인한 서비스 중지는 발생하지 않는다. 또한 여분서버의 재활은 해당 클러스터 시스템의 재활 주기에 따라 이뤄지므로 재활 후 클러스터 시스템의 불안정성은 제거된다. 마지막으로, (0,0) 상태는 클러스터 시스템의 고장 상태를 나타내며, 주서버와 여분서버의 가동이 모두 중지된 경우이다.

기존 연구에 정의한 가용도 분석은 클러스터 시스템의 한 서버라도 가동이 가능하면 가용하다고 분석하였다. 하지만 이와 같은 정의는 수학적인 분석으로는 의미가 있지만 현실적으로 사용자가 느끼는 가용도로 해석하기에는 문제가 있다. 따라서 본 논문에서는 서비스 계약 수준에 명시된 성능 척도를 만족시키기 위해 필요한 최소의 수(n)가 가동되지 못하면 시스템은 가용하지 않다고 정의한다. 또한 재활을 수행하는 동안에도 처리하던 서비스나 대기하던 요청에 대한 유실은 없지만 성능 척도를 고려하면 가용하지 않은 상태이다. 따라서 가동 가능한 서버수가 n 이상일 때만 클러스터 시스템이 가용함으로 차별화된 서비스 제공을 위한 클러스터 시스템의 가용도는 아래와 같이 정의된다.

$$A = \sum_{i=0}^{k} (P_{n,i} + P_{U_{n,i}}) \quad (1)$$

클러스터 시스템에서 제공 가능한 총 서비스 연결수를 성능 척도로 고려하여 클러스터 시스템의 구성을 실시간으로 변경할 수 있는 자체 구성 메커니즘을 제안한다. 또한 서비스 수준 계약을 만족하기 위해서 서버 처리 용량, 메모리, 네트워크 대역폭, 저장장치 등 다양한 자원 관리가 필요하지만 본 논문에서는 서버 처리 용량 중 제공 가능한 최대 서비스 연결수를 성능 척도로 고려한다. 왜냐하면 기존 여러 연구에서 증명하였듯이 서버 부하를 측정하기 위한 매개 변수 중 최대 서비스 연결수가 대표적인 지시자(indicator)가 되기 때문이다[4]. 그리고 성능 분석을 위해 서비스 도착률(α)과 처리 시간(β)은 상호 독립적이며, 지수 분포를 따른다고 가정한다. 먼저 서비스 수준 계약에서 정의한 지연 대기(waiting time)의 마감 시간(deadline)을 만족하기 위해 필요한 최소 서버수(m_0)를 결정하는데, 이것은 위의 가정을 따르면 Erlang's C formula, $C(m_0, \alpha/\beta)$[5]에 얻은 평균 지연 대기 함수로부터 얻을 수 있다.

$$W_q(t) = \frac{\left(\frac{(m_0\rho)^{m_0}}{m_0!}\right)\frac{1}{1-\rho}}{\left[\sum_{k=0}^{m_0-1}\frac{(m_0\rho)^k}{k!} + \left(\frac{(m_0\rho)^{m_0}}{m_0!}\right)\frac{1}{1-\rho}\right]} \quad (2)$$

서버수가 m_0일 때 임의의 대기 시간을 나타내는 변수를 W_q, 서비스 수준 계약서에 명시된 지연 대기 마감 시간을 d, 지연 대기 마감 시간을 만족하는 비율(Ψ)이라고 하면 사용자 클래스별 요구사항에 맞게 클러스터 시스템을 구성하기 위한 조건은 아래와 같다.

$$W(d) = P[W_q \le d] \ge \psi \quad (3)$$

3. 차별화된 서비스 제공을 위한 메커니즘

차별화된 서비스 제공을 위한 2 계층 클러스터 시스템의 초기 구성과 서비스를 시작하면서 적응적으로 초기 구성을 변경할 수 있는 메커니즘을 제안한다. 먼저 서비스 제공자는 서비스를 시작하기 이전에 사용자의 요구사항과 성향을 분석하여 클러스터 시스템의 초기 구성을 수행한다(initialize_configuration() 함수 참조). initialize_configuretion() 함수는 식 1에서 정의한 가용도 척도와 식 3에서 정의한 성능 척도를 고려한 것이다. 다음으로 서비스를 시작하면서 주기적으로 정적 서비스와 동적 서비스의 요청률을 분석하여 하나의 서비스만 집중되어 다른 서비스를 처리하기 위한 클러스터 시스템의 서버가 휴지 상태로 있거나, 일정 서비스를 동일 클러스터 시스템의 다른 서버가 작업을 인계해도 서비스 수준 계약 준수를 할 수 있다면 다른 서비스를 제공하기 위해 서버를 다른 클러스터 시스템으로 재구성한다(reconfigure()함수 참조). 또한 양 클러스터 시스템 모두 부하가 많이 걸렸을 경우 서비스 클래스에 우선순위를 두어 한 서비스가 서비스 수준 계약을 일정 기간동안 만족하지 못하더라도 우선순위가 높은 클래스의 서비스의 계약을 먼저 준수하기 위해 시스템 재구성을 수행한다. 이를 수식화하기 위해 정적 서비스가 우선순위가 높다고 하면 t 시간에 정적 서비스를 제공하기 위한 클러스터 시스템의 서버수를 S(t), 가용도를 $A_s(t)$, 한 서버의 최대 서비스 연결수를

$M_s(t)$라 하고, $t-1$ 시간에 총 서버 부하를 $L_s(t-1)$라고 하며, 가용도 요구사항을 Ω라고 하면 아래와 같은 조건을 만족할 때 서버수를 조정하며, 물론 동적 서비스의 경우에도 같은 원리로 관리되며 메커니즘을 장착하기 위한 의사코드는 그림 1과 같다.

$L_s(t) > S(t) \times M_s(t-1)$ or $A_s(t) < Ω$ 이면 $S(t) = S(t-1) + 1$

```
int initialize_configuration(double , double , int m0){
    While( equation (3) is not fulfilled ){
        Increment m0;
        calculate C(m0, /);
    }

    While ( availability requirement is not fulfilled ){
        Increment m0;
        calculate availability from equation (1);
    }
    return m0;
}

void reconfigure(){
   1= current sample arrival rate of static service;
   2= current sample arrival rate of dynamic service;
    S = current number of server in vlan1;
    D = current number of server in vlan1;

    if( > ){
        temp = initialize_configuration(, , S);
        D -= (temp - S);
         S = temp;
    }

    if( < ){
        temp = initialize_configuration(, , D);
        S -= (temp - D);
         D = temp;
    }
}
```

그림 1 차별화된 서비스 제공을 위한 메커니즘

3. 민감도 분석 결과

본 장에서는 가용도 분석과 가용도가 성능에 미치는 영향에 대한 민감도 분석을 수행하며, 분석 결과는 그림 2, 3과 같다. 그림 2은 소프트웨어 재활 메커니즘을 장착한 클러스터 시스템에 사용 가능한 여분서버가 없는 경우 서버의 결함에 따른 성능 저하에 대한 결과이다. 서버에 결함이 발생하여 서비스를 제공하지 못하기 때문에 발생하는 클러스터 시스템의 전체 성능 저하률(degradation ratio)을 (현재 제공되는 서비스 연결수)/(총 서버수 × 한 서버에서 제공가능한 연결수)로 계산한다. 결과에서 보듯이 소프트웨어 재활 메커니즘을 사용해도 서버의 결함이 발생하거나, 주서버의 수가 적으면 재활을 수행하려는 서버의 작업을 인계받을 서버가 없으므로 서비스 중단이 발생하게 된다. 따라서 주서버의 결함 발생이 시스템의 전체 성능에 영향을 미치므로 여분서버의 필요성을 보여준다. 그림 3은 주서버(primary server) 수와 여분서버(k) 수의 변화에 따른 클러스터 시스템의 가용도 분석 결과이다. 여분서버 수가 하나도 없을 경우 가용도 요구사항을 만족하지 못하고, 클러스터 시스템을 구성하기 위한 소프트웨어가 안정적이지 못하여 서버 고장률이 높아질수록 여분서버를 가지고 있더라도 가용도는 떨어지게 된다. 하지만 여분서버로 2대 서버를 확보할 수 있다면 고장률이 크더라도 시스템 요구사항을 만족할 수 있다. 따라서 제공하기 위한 서비스를 유형별로 나눠 각각의 처리를 위한 클러스터 시스템을 구성하여 다른 시스템의 시스템 요구사항과 서비스 수준 계약에 영향을 미치지 않는 한도에서 클러스터 시스템 간에 작업전이를 하여 전체 시스템의 가용도 및 성능을 향상할 수 있다.

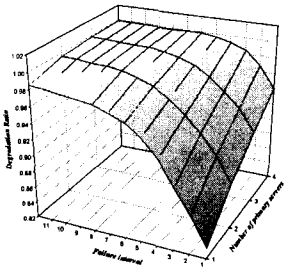

그림 2 시스템 성능 저하률

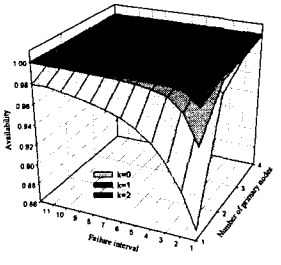

그림 3 여분서버수에 따른 가용도 분석

4. 결론

복잡한 소프트웨어가 필요한 응용분야에 기존에 보유하고 있는 컴퓨팅 자원을 사용하여 구축될 수 있는 클러스터 시스템을 적용할 수 있다. 하지만 제공되는 서비스의 질을 예측하여 가용성과 성능 요구사항을 만족하기 위한 메커니즘에 관한 연구가 필요하다. 따라서 본 논문에서는 차별화된 서비스를 제공하기 위한 메커니즘을 장착시킨 클러스터 시스템을 제안하고 이에 대한 민감도 분석을 수행하여 성능을 증명하였다. 또한 가용도와 성능 척도의 상관관계를 밝혀 목표로 했던 요구사항을 만족하기 위한 대안을 제시하였다.

참고문헌

[1] S. Garg, A. V. Moorsel, K. Vaidyanathan, and K. Trivedi, "A Methodology for Detection and Estimation of Software Aging," Proceedings of the 9th International Symposium on Software Reliability Engineering, pp. 282-292, Nov. 1998.

[2] B. Sericola, "Availability Analysis of Repairable Computer Systems and Stationary Detection," IEEE Transactions on Computers, Vol. 48, pp. 1166-1172, 1999.

[3] O. Pentakalos, D. Menasce, M. Halem, and Y. Yesha, "Analytical Performance Modeling of Hierachical Mass Storage Systems," IEEE Transactions on Computer, Vol. 42, No. 10, pp. 1103-1118, Oct. 1997.

[4] H. Chen and P. Mohapatra, "Session-Based Overload Control in QoS-Aware Web Servers," IEEE INFOCOM 2002, pp. 516-524, June 2002.

[5] L. Kleinrock, Queueing Systems Volume I: Theory, Wiley, 1975.