그리드 환경에서 동적 모니터링에 관한 연구

허의남[O] 김혜주 이웅재
서울여자대학교
{ huh[O], judy012, wjlee}@swu.ac.kr

An Dynamic Monitoring Interval for Grid Resource Information Services

Hye-ju Kim[O] Eui-Nam Huh woong-jae lee
Division of Information and Communication, Seoul Women's University

요 약

그리드 기술은 지리적으로 분산된 컴퓨팅 자원을 활용하기에 다양한 자원에 대한 정보를 관리하는 기술이 필요하며, 이는 자원 관리 시스템에서 자원을 모니터링하여 이루어진다. 자원 상태 정보가 실시간으로 변하여 주기적으로 모니터링을 행해야 한다. 모니터링에 사용된 자원은 시스템 성능에 오버헤드가 되므로 본 논문에서는 cpu 상태 변화 정도를 관찰하여 시스템 상태를 파악하고 이를 모니터링 주기에 적용시켜 모니터링 주기를 조절하는 시스템을 모델링 하였다. 이는 cpu상태 변화율에 따라 효율적이고 정확한 자원 정보를 수집하도록 모니터링 주기가 조절됨을 실험을 통하여 보일 것이다.

1. 서 론

그리드는 네트워크로 연결된 컴퓨팅 자원을 사용자가 개인용 컴퓨터를 이용하듯이 사용할 수 있다는 개념에서 출발한 네트워크로 연결된 가상의 슈퍼컴퓨터를 말한다.[1]

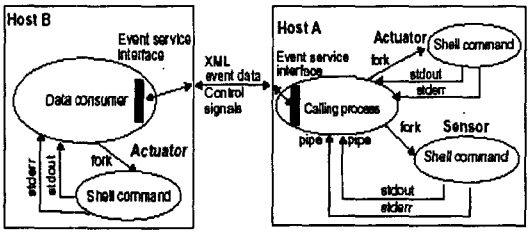
이기종의 컴퓨팅 자원과 대용량 저장장치, 다양한 고성능 연구 장비를 통합하는 그리드 환경에서 다양하고 많은 자원들은 그리드 자원 관리 시스템에 의해 관리되는데, 크게 자원 브로커와 정보서비스, 자원관리자로 구성되어 있다. 자원 브로커는 자원의 발견, 할당, 모니터링, 예약을 작업 단위로 자원 관리를 위한 모든 기능을 수행하고, 정보 서비스는 자원에 위치와 상태에 대한 정보를 수집하고 유지해 필요에 따라 제공해주는 역할을 한다. 그리고 자원관리자는 개별 자원에 대한 관리 기능을 담당한다. 이처럼 자원을 수집하고 자원의 정보를 파악하고 자원의 상태를 알기 위해서는 모니터링이 필요하다. 또한 그리드 상에서의 모니터링 기술은 오류가 감지된 응용프로그램, 네트워크, 호스트의 오류 혹은 overload가 감지될 경우 능동적으로 자원관리가 가능하도록 한다.[2][3]

실시간으로 변하는 그리드 자원의 특성상 정확한 자원 정보를 파악하기 위해 실시간 모니터링이 이루어져야 한다. 하지만 실시간으로 모니터링을 하는 것은 시스템 성능에 굉장한 오버헤드가 되므로 얼마나 효율적으로 모니터링을 하는가가 중요하다. 예를 들어 자원 상태 변화가 거의 없는데 짧은 주기로 모니터링이 계속 이루어진다면 모니터링에 의한 오버헤드만이 증가할 것이며, 자원 상태 변화가 큰데 긴 주기로 모니터링이 이루어진다면 정확하지 않은 자원 정보를 가지고 자원 관리 시스템에서 자원 관리를 하게 된다.

본 연구에서는 자원 상태 변화가 많은 그리드 환경의 특성을 고려하여 자원 상태의 변화 정도를 감시하여 모니터링 주기를 조절하고자 한다. 즉 자원 상태 변화가 큰 경우에는 모니터링 주기를 줄여 빠르게 자원 상태 정보를 알려주고, 자원 상태 변화가 크지 않을 경우에는 모니터링 주기를 길게 조절하는 것이다.

2. 관 련 연 구

Information Power Grid [IPG]는 NASA에서 그리드 중 계산 그리드를 위한 좀더 효율적인 모니터링을 지원하고자 만든 인프라이다. 계산 그리드의 특성상 자원의 규모가 크고 복잡하기에 오류를 유발할 수 있는 가능성이 높다. 그래서 잘못을 유발할 수 있는 조건을 찾도록 자원을 모니터링하는 메커니즘이 필요하고, 또한 QoS 조절하기 위한 회복 메커니즘도 요구되어 진다. 이러한 기능을 기반으로 디자인되어 있으며, 다음과 같이 제시하고 있다.[3]



[그림 1] 3가지 컴포넌트를 이용한 모니터링 시스템

세가지 기본적인 컴포넌트는 센서와 구동기, 이벤트 서비스로 구성되어 있다. 첫째 센서(sensor)는 타겟

시스템의 자원의 측정을 측정한다. 둘째 구동기(actuator)는 센서의 작업과 동일하지만, 다른 점은 특정화된 작업이나, 작업관리 또는 사용자 정의 작업을 수행하는데 shell script을 사용한다. 셋째는 이벤트 서비스(event service)는 센서에서 수집된 정보를 이 정보에 관심이 있는 다른 작업공간에게 제공하는 메커니즘이다. 또한 적용된 이벤트 정보를 사용자 작업에서 소비자 작업으로 전달한다.[4]

그러나 이 모니터링도 모니터링하면서 사용되어진 많은 자원에 대해서는 고려하지 않은 채 모니터링된 자료를 어떻게 잘 알릴 것인지에 대해서만이 언급되고 있다. 본 논문에서는 자원 상태 변화 정도를 파악해 모니터링 주기를 조절하여 효율적으로 모니터링하는 시스템을 제안한다.

3. 동적 모니터링 주기(Dynamic Monitoring Interval)

자원 정보 관리 시스템에서 자원 정보를 발견하면 리스트에 넣고, 자원을 할당하려 할 때 검색하거나 모니터링을 하게 된다. 이렇게 모니터링된 자원은 하드웨어와 소프트웨어 정보, 동적인 시스템 상태 정보이며, 이것으로 자원관리를 한다. 자원을 할당하고 관리할 때 중요시 고려하는 요소는 cpu사용률, 메모리 사용률, 너트워크 상태 정보 등이다. 그런데 그리드 프로그램 뿐만 아니라 분산 처리용 프로그램이 수행될 때는 cpu 변화에 따라 종속적으로 나머지 모니터링 요소들이 변하기 때문에 cpu 사용률을 파악하여 자원 상태 변화를 예측할 수 있다. 예를 들어, cpu의 사용률이 변화 없다면 그 나머지 이벤트들도 모니터링 할 필요가 없다. 즉 단지 cpu 사용률만으로 주기를 변경하는 것이 아니라 cpu 사용률의 변화율이 높고 낮음으로 모니터링 주기를 조절하자는 것이 본 논문의 제안이다. 즉, cpu 사용률의 변화가 큰 경우에는 자원 정보 상태의 변화 가능성이 큰 경우이므로 모니터링 주기를 짧게 해서 자원 정보를 자원 정보 관리 시스템에게 알리는 것이다. 여기서 자원 정보의 상태변화율은 이전의 모니터링으로 알아온 cpu 사용률과 현재 모니터링된 cpu 사용률의 차를 통해 알 수 있고, 이 차이가 크다는 것은 시스템 변화가 생겼다는 것을 의미해 모니터링 주기를 줄이게 된다.

cpu 사용률로 자원 정보 상태 변화를 알아보는 방법을 모델링을 하면, 현재 모니터링을 통해 들어오는 cpu 상태 정보는 현재 cpu가 사용하고 있는 cpu 사용률을 %로 표현한다. 이 때 현재 다른 자원에게 자원을 할당해 줄 수 있는 양을 100%에서 현재 사용량을 뺀 값으로 e(k)라고 하자. (e(k-1)은 전단계에서 모니터링된 cpu 사용률을 뺀 %가 된다.) 또 다른 변수로 현재의 자원 정보 변화 뿐만이 아니라 과거의 history까지 고려한 PID(Proportional Integrated Differential)컨트롤러를 이용한 dynamic threshold로 자원 정보 상태를 예측하는 u(k)를 정의한다.[5] PID 컨트롤러를 이용한 dynamic threshold로 u(k)를 표현해보면 [식 1]과 같다.

$$(k_p \times e(k) + k_i \times \sum_{j=1}^{k} e(j) + k_d \times (e(k) - e(k-1)) \quad [식\ 1]$$

이때 PID 컨트롤러는 각각 k_p =-0.5, k_i=0.125, k_d =-0.125로 조율을 맞춘다. 즉 u(k)을 PID 컨트롤러를 이용한 현재 모니터링된 cpu idle 값인 e(k)로 나타낼 수 있고 cpu사용률의 변화량은 |u(k)-u(k-1)|를 통해 알 수 있다.

|u(k)-u(k-1)|의 값을 간단하게 정리해 보면, k_i*e(k)결과 값과 비교해 볼 수 있는데, |u(k)-u(k-1)|이 k_i*e(k)보다 작거나 같으면 cpu사용률의 변화량이 거의 없는 안정된 상태를 의미한다. 하지만 |u(k)-u(k-1)|가 k_i*e(k)보다 크다면 자원 정보의 상태가 변화한 것을 의미한다. 그러므로 |u(k)-u(k-1)|값의 변화와 k_i*e(k)값의 변화를 비교해 자원 상태 변화를 파악해 예측할 수 있다. [6]

4. 실 험 및 결 과

본 절에서 나온 [식 1]을 자원 정보 상태를 2가지에 적용시켜 보겠다.

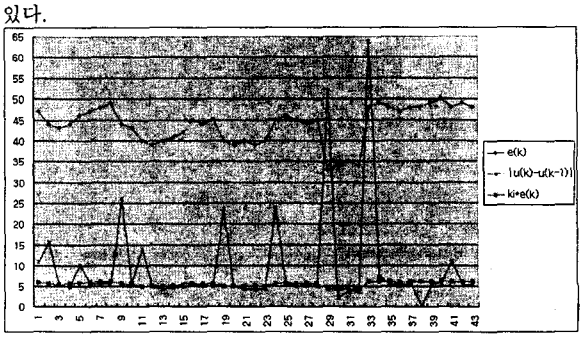
첫번째 적용 환경은 자원 정보 변화 상태가 크지 않는 smooth한 경우이다. 이 경우에는 자원 정보 변화가 작더라도 변한 상태이므로 이를 모니터링 주기에 반영하여 모니터링 주기를 조절할 필요가 있다.

[표 1] 시스템 변화율이 smooth한 경우

smooth monitor

cpu사용률	e(k)	u(k)	u(k-1)	|u(k)-u(k-1)|	ki-e(k)
53	47	117.75	128.75	11	5.875
57	43	96.875	102.125	5.25	5.375
56	44	101.625	96.875	4.75	5.5
54	46	111.875	101.625	10.25	5.75
53	47	117.375	111.875	5.5	5.875
52	48	122.875	117.375	5.5	6
51	49	128.5	122.875	5.625	6.125
61	39	78.125	82.876	4.75	4.875
60	40	82.375	78.125	4.25	5
59	41	87	82.375	4.625	5.125
58	42	91.75	87	4.75	5.25
53	47	116.876	91.75	25.125	5.875
54	46	112.25	116.875	4.625	5.75
55	45	107	112.25	5.25	5.625
56	44	101.875	107	5.125	5.5
55	45	106.75	101.875	4.875	5.625
60	40	83.125	106.75	23.625	5
61	39	78.125	83.125	5	4.875
60	40	82.375	78.125	4.25	5
61	39	78.125	82.375	4.25	4.875
60	40	82.375	78.125	4.25	5
55	45	106.25	82.375	23.875	5.625
54	46	112	106.25	5.75	5.75
55	45	107	112	5	5.625
56	44	101.875	107	5.125	5.5
55	45	106.75	101.875	4.875	5.625
59	34	57.25	55.125	2.125	4.25
66	35	61.125	57.25	3.875	4.375
66	34	57.5	61.125	3.625	4.25

위의 표를 분석해보면, 자원 정보 변화 정도가 1에서 5정도로 작은 경우, 1,2정도의 변화보다 4,5정도의 변화가 크므로 그 변화를 알아야 하는데, 이때 사용 가능한 자원 정보 변화가 갑자기 40에서 45로 늘어난 경우 |u(k)-u(k-1)|의 값이 k_i*e(k)보다 크므로 자원 정보 상태 변화가 있는 것이므로 모니터링 주기를 줄이도록 조절한다. 위의 표를 간단하게 차트화를 시키면, |u(k)-u(k-1)|값이 갑자기 변화한 경우 k_i*e(k)의 그래프도 변화가 크게 나타난다는 것을 알 수 있다.



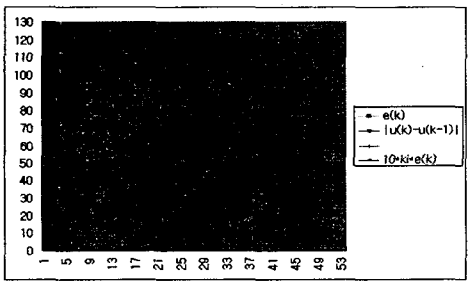
[그림 2] smooth한 시스템 변화에서의 변화가능성

두 번째 적용환경은 자원 정보 상태가 굉장히 많이 변화하는 경우이다. 예를 들어 자원의 정보가 계속해서 크게 5에서 20정도로 변화 할 경우, 앞의 환경보다 변화가 크다고 모니터링 주기를 짧게 하여 체크하면 overhead가 크게 된다. 이때는 지금 현 상황에서 cpu 사용률의 작은 변화와 큰 변화를 파악하여 자원 정보 상태가 변화했음을 체크 해야 한다.

[표 2] 시스템 변화율이 dynamic한 경우

monitor					
cpu사용률	e(k)	u(k)	u(k-1)	|u(k)-u(k-1)|	ki*e(k)
58	42	113.5	85.25	28.25	5.25
55	45	141.125	113.5	27.625	5.625
80	20	76.8125	141.125	64.3125	2.5
78	22	48.03125	76.8125	28.78125	2.75
77	23	35.51563	48.03125	12.51563	2.875
62	38	72.38281	35.51563	36.86718	4.75
63	37	87.06641	72.38281	14.6836	4.625
61	39	102.0332	87.06641	14.96679	4.875
60	40	113.5166	102.0332	11.4834	5
73	27	77.0083	113.5166	36.5083	3.375
72	28	61.25415	77.0083	15.75415	3.5
74	26	48.50208	61.25415	12.75207	3.25
73	27	44.50104	48.50208	4.001042	3.375
71	29	47.62552	44.50104	3.124479	3.625
74	26	41.68776	47.62552	5.937761	3.25
77	23	32.34388	41.68776	9.34388	2.875
55	45	100.5469	32.34388	68.20306	5.625
56	44	130.0235	100.5469	29.47657	5.5
52	48	164.0117	130.0235	33.98823	6
53	47	176.0059	164.0117	11.99417	5.875
54	46	177.1279	176.0059	1.122034	5.75
40	60	257.314	177.1279	80.18607	7.5
41	59	290.907	257.314	33.59298	7.375
39	61	320.8285	290.907	29.92149	7.625
38	62	342.5392	320.8285	21.71075	7.75
37	63	360.2696	342.5392	17.73042	7.875

위의 표를 분석해보면, 자원 정보 변화 정도가 |u(k)-u(k-1)|의 값이 k_i*e(k)보다 작은 경우는 자원 상태 변화가 작으므로 모니터링 주기를 길게 하고, 큰 경우에는 자원 상태 변화가 크므로 모니터링 주기를 짧게 하는 것이다. 예를 들어 자원 정보 사용 가능한 정도가 46%에서 60%로 증가하면 자원 정보 상태가 변한 것으로 |u(k)-u(k-1)|의 값이 k_i*e(k)보다 크게 된다. 위 표를 차트화 하면 다음과 같다.



[그림 3]dynamic한 시스템 변화에서의 변화가능성

|u(k)-u(k-1)|값이 변화가 굉장히 큰 경우에서도 더 큰 변화에 뾰족하게 나타나는 순간이 있다. 이때 |u(k)-u(k-1)|값이 k_i*e(k)보다 크게 되어 자원 정보 상태의 변화를 파악하여 모니터링 주기를 짧아지도록 적용시킨다.

특정 환경에 5% threshold로 변화를 detect했을 경우 100번의 모니터링을 수행했을 때, 본 연구를 통해 모니터링의 오버헤드를 20%정도 줄이는 효과를 볼 수 있다.

5. 결 론

그리드 환경에서 자원 관리 시스템이 자원의 정보를 분석하고 관리하기 위해서는 반드시 모니터링 과정이 선행되어야 한다. 또한 효율적인 자원관리를 위해서도 자원 정보에 관한 모니터링은 반드시 이루어져야 한다. 여기서 문제는 반드시 이루어져야 하는 자원 정보에 대한 모니터링을 얼마나 자주 하느냐 하는 것이다.

그리드 환경은 굉장히 유동성이 크며, 실시간으로 달라질 가능성을 갖고 있기에 일정한 주기를 주어 자원 정보에 대한 모니터링을 해오는 기존의 방식이 문제가 있다. 다시 말해서 그리드 환경의 자원 성능 상태가 어떠한지도 모르고 모니터링을 하는 것이다.

본 연구는 자원 정보 상태의 변화 가능성을 예측하기 위해 자원 정보 모니터링에서 얻어온 cpu 정보와 PID 컨트롤러를 이용한 |u(k)-u(k-1)|값과 k_i*e(k)을 비교 하였다. 이 비교를 통해 자원 정보 상태 변화를 파악하고, 이를 이용해 모니터링 주기를 조절하여 최소의 모니터링 overhead로 정확한 자원 정보를 얻도록 하는 것이다. 시스템의 상태 변화가 많은 환경과 그렇지 않은 환경에 적용해 이를 증명하였다.

또한 본 논문에서 제안된 모델링을 현재 그리드 모니터링 기반으로 설계되어 새로운 하나의 모듈을 추가함으로써 모니터링 주기를 자원 성능 상태에 따라 다르게 할 수 있다. 즉 본 논문을 적용하는 면에 있어서도 현재의 구성요소와 다른 코드에 변화를 주지 않기에 쉽게 최적화된 모니터링 주기를 얻을 수 있을 것이다.

참 고 문 헌

1. I. Foster., "The Grid: A New Infrastructure for 21st Century Science.", *Physics Today*, 55(2):42~47 2002
2. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B, Schwartzkopf, H, Stockinger, K. Stockinger, B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services.", *Proceedings of Supercomputing 2002 (SC2002)*, November 2002
3. Tierney, B., B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson "A Monitoring Sensor Management System for Grid Environments" *Proceedings of the IEEE High Performance Distributed Computing conference* (HPDC-9), August 2000, LBNL-45260
4. A. Waheed, W. Smith, J. George, J. Yan. , "An Infrastructure for Monitoring and Management in Computational Grids", In Proceedings of the 2000 Conference on Languages, Compilers, and Runtime Systems.
5. Ashvin Goel, David Steere, Calton Pu, and Jonathan Walpole, "Adaptive Resource Management Via Modular Feedback Control", Submitted to HOTOS 1999
6. PID Tutorial, http://www.engin.umich.edu /group /ctm /PID/PID.html