고가용성 시스템을 위한 최적의 Heartbeat 간격에 대한 연구

박주용[0] 김재훈
아주대학교 정보통신전문대학원
{ex94co,jaikim}@dmc.ajou.ac.kr

A Study on the Optimal Heartbeat Intervals for High Availability Systems

Joo-Yong Park[0] *Jai-Hoon Kim*
The Graduate School of Information and Communication
Ajou University

요 약

일반 시스템에서는 결함이 발생하였을 때, 즉 어떠한 작업을 수행하는 프로세스 또는 하드웨어에 결함이 발생하였을 때 작업이 중단되거나 처음부터 다시 수행하여야 한다. 그러나, 고가용성 시스템은 일반 다른 시스템과 달리 어떠한 결함이 발생했을 때에도 지속적으로 서비스를 수행할 수 있는 하드웨어나 소프트웨어 시스템이 구축되어 있다. 고가용성 시스템에서 Heartbeat 을 이용하여 시스템에서 발생하는 결함을 발견하여 필요한 조치를 취할 수 있도록 한다. 또한, 체크포인트(Checkpoint)와 롤백(Roll-Back) 기법을 사용하여 컴퓨팅의 손실을 최소화하기 위하여 컴퓨팅 작업을 처음부터 다시 시작하는 것이 아니라 최근의 상태 저장 순간으로 되돌아가 다시 시작한다. 본 논문에서는 고가용성 시스템에서 체크포인트와 Heartbeat 을 사용할 때 체크포인트 간격과 Heartbeat 간격에 따른 평균 수행시간을 구하고, 최적의 체크포인트 간격을 적용한 경우에 Heartbeat 간격에 따른 시스템의 성능을 분석하고 비교하였다.

1. 서론

고가용성(High-Availability)이란 하드웨어나 소프트웨어, 네트워크등의 자원에 문제가 발생했을 때에도 지속적으로 서비스를 제공하는 기술을 말한다[7]. 고가용성 시스템에서는 결함이 발생 했을 때에 지속적인 서비스를 제공하기 위하여 여러 서비스 중단 요인들에 대한 fail-over 기능이 필요하다[4,5]. Heartbeat[6]과 체크포인트(Checkpoint) 와 롤백(Roll-Back) 기법[1]을 사용하여 결함이 일어났을 때에 이를 감지하고 컴퓨팅의 손실을 최소화할 수 있다.

체크포인트는 안정된 저장장치(예: 하드디스크)에 어플리케이션의 상태를 저장한다. 결함이 발생하면 안정된 저장장치에 저장된 체크포인트 시점으로 롤백하여 다시 진행한다[1]. 그러나, 체크포인트와 롤백 기법은 일시적인 결함(transient fault)이 발생 했을 때에 에러 복구 기법으로 사용해 왔다. 체크포인트와 롤백 기법만으로는 영구적인 결함(permanent fault)이 발생 했을 때 에러를 복구 할 수 없다. 그렇기 때문에 고가용성 시스템에서 체크포인트와 롤백 기법과 물리적인 중복기법을 함께 사용하여 영구적인 결함이 발생했을 때에도 지속적인 서비스를 수행할 수 있다. 물리적인 중복의 경우 노드들의 상태를 주기적으로 저장할 필요가 있다.

Heartbeat은 노드들간의 작업수행 가능여부를 주기적으로 체크하고, 만약에 서비스 수행도중에 결함이 발생하면 고가용성 시스템의 대기 노드는 Heartbeat메시지를 통하여 현재 프로그램 수행중인 노드의 결함 발생을 인식하고, 공유 디스크에 저장된 가장 최근의 체크포인트로 롤백하여 서비스를 계속 수행한다.

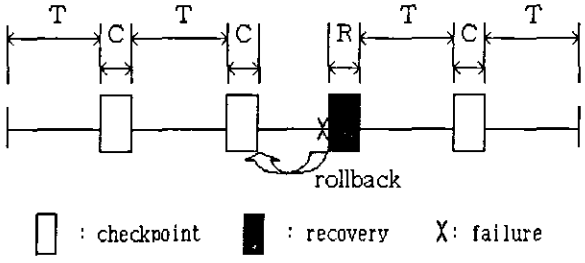
본 논문은 고가용성 시스템에서 체크포인트와 Heart-beat 을 사용할 때 체크포인트 간격과 Heartbeat 간격에 따른 실행 시간의 기대값을 구하고, 최적의 체크포인트 간격[1]을 적용한 경우에 Heartbeat 간격에 따른 시스템의 성능을 분석하였다.

2. 관련연구

고가용성 시스템은 어떠한 문제가 발생하더라도 서비스를 지속적으로 수행 할 수 있어야 한다. 그러기 위해서는 현재 서비스를 수행하고 있는 노드가 지속적으로 작업을 수행 할 수 있는지에 대하여 주기적으로 확인 하여야 한다. 그 때에 쓰이는 기술이 Heartbeat이다. Heartbeat은 고가용성 시스템내의 결함여부를 상호간 체크하여 필요한 조치를 취하도록 한다[6]. Heartbeat 에는 시스템 점검기능과 IP takeover 기능이 있으며, Heartbeat 메시지는 시리얼라인, UDP,

PPP/UDP 등을 통하여 전송 될 수 있다. 만약 메시지가 전송되지 않으면, timeout 카운트가 시작되고, 계속해서 얼마 동안 Heartbeat 메시지를 받지 못하면, 노드는 장애가 발생한 것으로 간주한다[7]. 각 노드들은 주기적으로 Heartbeat를 주고 받으며 서로의 상태를 확인한다. 그때 서비스를 수행 중이던 노드가 결함이 발생하였다고 판단되면 대기중인 노드가 서비스를 수행한다. 결함발생시에 컴퓨팅의 손실을 최소화하기 위하여 체크포인트와 롤백 기법을 사용한다. 서비스 중인 노드는 일정한 주기로 체크포인팅하여 체크포인트까지의 작업상태를 안전한 공유 디스크에 저장한다. 결함이 발생하면 대기중이던 노드는 안전한 공유디스크에 저장된 체크포인트 시점으로 롤백하여 그 시점부터 서비스를 다시 수행 한다.

체크포인트와 롤백 기법에서 체크포인트 간격이 줄어들수록 롤백 간격도 줄어들 수 있지만, 체크포인팅 과정에서 비용이 소요되어 전체적인 성능을 저해할 수 있는 요소가 된다. 이렇기 때문에 적절한 체크포인트 간격이 요구된다[1,3].



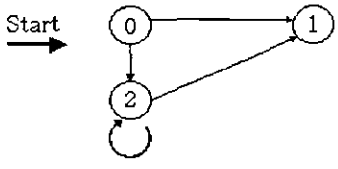
[그림 1] 체크포인트와 롤백 기법

[그림 1]은 결함 발생시 체크포인트와 롤백에 대한 과정을 나타낸다. T는 체크포인트 간격을 나타내고, C는 체크포인트 오버헤드, R은 가장 최근의 체크포인트로의 롤백 오버헤드, λ 는 오류 발생률이라 할 때 최적의 체크포인트 간격 T_{opt}는 다음과 같다[1].

$$T_{opt} = \sqrt{\frac{2C}{\lambda}}$$

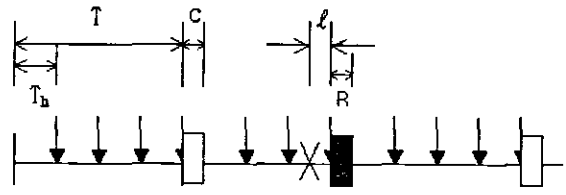
3. 분석

체크포인트와 Heartbeat 을 사용한 고가용성 시스템에서 최적의 체크포인트 간격 내에서의 Heartbeat 간격이 컴퓨팅 성능에 어떠한 영향을 미치는지 분석하면 다음과 같다.



[그림 2] Markov chain

Markov chain에서 X 상태에서 Y 상태로의 전환에서 전환확률은 P_{XY}, 비용은 K_{XY} 라 한다. T는 체크포인트 간격, C는 체크포인트 오버헤드, R은 가장 최근의 체크포인트로의 롤백 오버헤드, λ는 오류 발생률, T_h 는 Heartbeat



[그림 3] 체크포인트와 롤백 기법과 Heartbeat

간격, C_h 는 Heartbeat 오버헤드를 나타낸다.
n 은 T 내에서의 T_h의 횟수이며 T 간격 내에서 T_h 는 정수개 존재한다고 가정하였다. 따라서 $0 < T_h < T$ 이고, $n*T_h = T$ 한다.

$$P_{01} = e^{-\lambda(T+C+n*C_h)}$$

$$K_{01} = T + C + n*C_h$$

$$P_{02} = 1 - P_{01} = 1 - e^{-\lambda(T+C+n*C_h)}$$

$$K_{02} = \int_0^{T+C+n*C_h} (t)\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda(T+C+n*C_h)}}dt + (T_h + C_h) - \int_0^{T_h+C_h} (t)\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda(T_h+C_h)}}dt$$

$$= -\frac{(T+C+n*C_h)e^{-\lambda(T+C+n*C_h)}}{1-e^{-\lambda(T+C+n*C_h)}} + (T_h + C_h) + \frac{(T_h+C_h)e^{-\lambda(T_h+C_h)}}{1-e^{-\lambda(T_h+C_h)}}$$

$$P_{21} = e^{-\lambda(R+T+C+n*C_h)}$$

$$K_{21} = R + T + C + n*C_h$$

$$P_{22} = 1 - P_{21} = 1 - e^{-\lambda(R+T+C+n*C_h)}$$

$$K_{22} = \int_0^{R+T+C+n*C_h} (t)\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda(R+T+C+n*C_h)}}dt + (T_h + C_h) - \int_0^{T_h+C_h} (t)\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda(T_h+C_h)}}dt$$

$$= -\frac{(R+T+C+n*C_h)e^{-\lambda(R+T+C+n*C_h)}}{1-e^{-\lambda(R+T+C+n*C_h)}} + (T_h + C_h) + \frac{(T_h+C_h)e^{-\lambda(T_h+C_h)}}{1-e^{-\lambda(T_h+C_h)}}$$

Heartbeat 주기내에서 결함이 발생 하였을 때, Heartbeat 메시지가 도착하기 전까지는 결함이 발생하였음을 알지 못한다. 따라서 Heartbeat 을 통하여 결함이 발생했음을 인식하고, 체크포인트까지 롤 l하여 다시 서비스를 수행할 수 있다. [그림 3]에서 ℓ 는 결함발생 시점에서부터 Heartbeat 메시지가 도착할 때까지의 대기상태에 소요되는 평균 비용이다.

ℓ 는 $(T_h + C_h) - \int_0^{T_h+C_h} (t)\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda(T_h+C_h)}}dt$ 으로 표시할 수 있다. K_{02} 는 결함발생시점까지의 비용과 ℓ 의 합으로 나타내었다.K_{22} 또한 같은 방법으로 계산하였다.

평균 실행 시간 Γ는 상태 0에서 상태 1로 전환되는데 소요되는 비용이다. 다음과 같은 식으로 나타낼 수 있다.

$$\Gamma = P_{01}K_{01} + P_{02}(K_{02} + \frac{P_{22}}{1-P_{22}}K_{22} + K_{21})$$

위 의식을 전환 확률과 비용의 식으로 치환하여 정리한 것이 아래의 식이다.

$$\Gamma = (e^{\lambda(R+T+C+n*C_h)} - e^{\lambda R}) * \frac{(T_h + C_h)e^{\lambda(T_h+C_h)}}{e^{\lambda(T_h+C_h)} - 1}$$

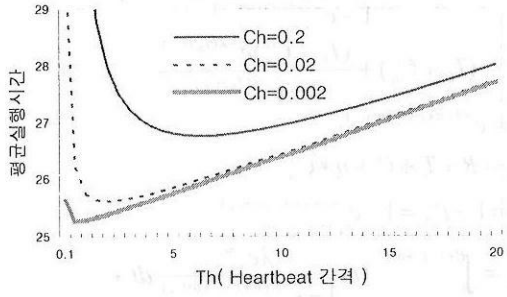
4. 성능분석

최적의 체크포인트 간격 $T_{opt} = \sqrt{\frac{2C}{\lambda}}$ 를 사용하였을 때 Heartbeat 주기에 따른 평균 실행시간 Γ 값을 구하였고 최소의 실행시간을 위한 최적의 Heartbeat 간격 T_{h_opt} 값을 구하였다.

체크포인트로의 롤백 오버헤드 (R = 2), 체크포인트 오버헤드 (C = 2), 오류발생률(λ =0.01), 최적의 체크포인트 간격 (T_{opt} =20)일때,

$$\Gamma = (e^{\lambda(R+T+C+n*C_h)} - e^{\lambda R}) * \frac{(T_h + C_h)e^{\lambda(T_h+C_h)}}{e^{\lambda(T_h+C_h)} - 1}$$

에서 Heartbeat 오버헤드 (C_h) 의 변화에 대한 Heartbeat 간격 (T_h) 변화에 따른 Γ의 결과를 [그림 4]에서 나타내었다. (단, $0<T_h<T_{opt}$)



[그림 4] Heartbeat 간격에 따른 평균실행시간

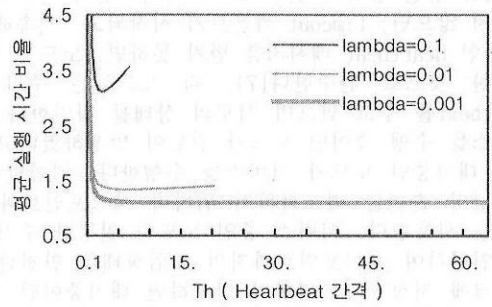
C_h	T_{h_opt}
0.2	6.3
0.02	2
0.002	0.63

[표 1] Heartbeat 오버헤드(C_h)에 따른 최적의 Heartbeat 간격(T_h)

[그림 4]에서 볼 수 있듯이 최적의 체크포인트 간격 내에서 최적의 T_h 가 존재함을 알 수 있다. 또한, Heartbeat의 오버헤드 C_h 값이 작아 짐에 따라서 T_h 의 값도 작아짐을 알 수 있다. 그리고, C_h 의 값이 작아질수록 실행시간이 줄어드는 것을 알 수 있다. [표 1]은 Heartbeat 오버헤드(C_h)에 따른 최적의 Heartbeat 간격(T_{h_opt})을 나타낸다.

[그림 5]는 R = 2, C = 2, C_h = 0.02 일 때의T_h 에 따른 평균실행시간비율 ($\frac{\Gamma}{T_{opt}}$)을 나타낸 그래프이다.

(단, $0<T_h<T_{opt}$)



[그림 5] Heartbeat 간격에 따른 평균실행시간비율

[그림 5]에서 볼 수 있듯이 λ의 값이 작아질수록 완만한 곡선을 이루는 것을 알 수 있다. 이것은 오류 발생률이 작을수록 Heartbeat 간격이 실행 시간에 큰 영향을 주지 않음을 보여준다. 다시 말하면 오류발생률에 따른 적절한 Heartbeat 간격이 필요로 함을 알 수가 있다. [표 2]는 오류발생률에 따른 최적의 체크포인트 간격(T_{opt})와 최적의 Heartbeat 간격(T_{h_opt})을 나타낸다.

λ	T_{opt}	T_{h_opt}
0.1	6.324555	0.67
0.01	20	2
0.001	63.24555	6.3

[표 2] 오류발생률에 따른 최적의 체크포인트 간격(T_{opt})와 최적의 Heartbeat 간격(T_{h_opt})

5. 결론 및 향후과제

고가용성 시스템에서의 컴퓨팅 손실을 최소화하기 위하여 체크포인트와 롤백 기법을 사용하고, 각 노드들 간에 결함 발생을 확인 하기 위하여 Heartbeat 을 사용하였을 때 최적의 체크포인트 간격 내에서의 최적의 Heartbeat 간격이 존재함을 알 수 있었다. 이때 최적의 Heartbeat 간격이 Heartbeat 오버헤드와 결함 발생 확률과 관계가 있음을 알 수 있었다. 이러한 최적의 Heartbeat 간격을 실험적으로 구하였는데 수학적으로 구하는 연구를 진행중이다.

6. 참고 문헌

[1] N.H.Vaidya, " On Checkpoint Latency," *in Proc. of the 1995 Pacific Rim International Symposium on Fault-Tolerant Systems*, pp.60-65, Dec.1995.

[2] Jai-Hoon Kim and N.H.Vaidya, " Analysis of one-level and two-level failure recovery schemes for distributed shared memory system," IEE Proceedings-Computers and Digital Techniques, Vol.146, Issue 3,pp.125-130, May.1999.

[3] J.Young " A first order approximation to the optimal checkpoint interval," *Communication of the Acm, Vol.17 pp.530-531, Sept.1974*

[4] Linux High Availability HOWTO, http://halinux.rug.ac.be/High-Availability-HOWTO-5.html

[5] High Availability Linux Project, http://linux-ha.org/

[6] Heart redundant, distributed cluster technology, http://www.lemuria.org/Heart/

[7] 최재영 외, " 고가용성 리눅스," 정보처리 제6권 6호, pp.19-25, 1999.