논문 18-43-02-12 The Journal of Korean Institute of Communications and Information Sciences '18-02 Vol.43 No.02
https://doi.org/10.7840/kics.2018.43.2.284

가상화 환경에서의 동적 체크포인트 주기 결정 알고리즘

이 한 범◆, 김 화 성°

Dynamic Checkpoint Period Decision Algorithm in Virtualized Environments

Han-bum Lee◆, Hwa-sung Kim°

요 약

가상화 환경에서 끊김 없는 서비스 제공을 보장하기 위해서는 하드웨어의 견고함과 동시에 체크포인트와 같은 가상화 된 자원의 장애 관리 메커니즘이 요구된다. 기존 체크포인트 방식들은 프로세스 별 장애 발생 확률을 고려하여 장애가 발생 하여도 수행 시간을 최대한 보장하기 위한 체크포인트 주기를 결정한다. 이러한 방식들은 고정 주기의 체크포인트를 수행하여 상대적으로 오버헤드가 적지만, 실시간 상태를 고려한 주기 변경을 할 수 없기 때문에 장애 대처가 느린 문제가 있다. 본 논문은 장애 관리의 일환으로 가상자원을 모니터링 하여 실시간 상태에 따라 체크포인트 주기를 동적으로 변경하는 방법을 제안한다.

Key Words : Checkpoint, Virtualization, Fault management, monitoring, container

ABSTRACT

In order to ensure seamless service provisioning in virtualized environments, a fault management mechanism of virtualized resource is also required as well as the robust hardware. Existing checkpointing methods determine a checkpoint period that maximizes the execution time even if a failure occurs considering the probability of task failure. In these schemes, the overhead of the system is relatively low, but it is difficult to manage faults quickly because the period can not be changed considering the real time state. In this paper, we propose a method to change checkpoint period according to real-time status.

Ⅰ. 서 론

가상화 기술은 고정 된 하드웨어 공간에 분리 된 가상자원을 만들어 운영체제나 특정 하드웨어 구조의 영향을 받지 않고 별도의 실행 환경을 운영 가능토록 한다. 가상화 된 자원에서 운용하는 서비스들의 고가용성을 보장하기 위해서는 하드웨어 이중화를 통해 여분의 자원을 운용할 수 있지만 이는 서버 관리에 추가적인 부담을 줄 수 있다. 따라서 가상화 환경에서는 체크포인트 등의 소프트웨어적인 방식의 장애 관리 기법들이 추가로 요구 된다.

체크포인트 방식은 프로세스의 상태를 일정 주기로 저장하여 장애 발생 시 이전 상태로 돌아가 장애를 회복하도록 하는 기술이다. 고전적인 체크포인트 방식은 프로세스의 장애 발생율을 측정하여 장애 위치를 예측 후 프로세스의 수행 시간을 최대화 하는 최적의 체크포인트 주기를 계산한다[1]. 하지만 가상화 환경에서 각 서버는 여러 프로세스를 함께 관리하기 때문에 장애 발생율과 같이, 실행 후 알 수 있는 파라미터들을 미리 알고 주기결정에 반영하기 힘들다. 따라서 각 프로세스 별 체크포인트가 아닌 가상자원 별 체크포인트 주기를 설정하는 방식이 요구된다.

Ⅱ. 고정 주기를 활용한 체크포인트 방식

체크포인트의 주기가 짧으면 장애 발생 시점과 마지막 상태 저장 시점이 가까울 확률이 높기 때문에 손실되는 작업량이 적지만 체크포인트 동작 자체에 오버헤드가 커지게 된다. 하드웨어 시스템에서 가장 대

※ 이 논문은 2016년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신연구기반구축사업의 일환으로 수행하였음. [12221-14-1001, 차세대 네트워크·컴퓨팅 플랫폼연구 기반구축]
◆ First Author : (0000-0003-1056-5710)Kwangwoon University Department of Electronics and Communications Engineering, krrcby@kw.ac.kr, 학생회원
° Corresponding Author : (0000-0001-5893-5691)Kwangwoon University Department of Electronics and Communications Engineering, hwkim@kw.ac.kr, 종신회원
논문번호 : KICS2018-01-022, Received January 19, 2018; Revised January 30, 2018; Accepted January 30, 2018

표적인 체크포인트 주기 연구는 Young의 일차 근사법을 활용한 고정 주기 도출법이 있다[2]. Young은 Poisson 분포를 따르는 과도 장애가 발생 가능한 상황에서 체크포인트 주기와 시스템 다운타임의 관계식을 정의하고, 이를 최소화 하는 주기 도출식을 도출하였다. 이 때 체크포인트 주기는 식 (1)과 같은 값을 갖는다. T_c는 체크포인트의 주기, T_s 는 체크포인트 시 소모되는 저장 시간, T_f는 장애 간 평균 간격 시간을 나타낸다.

$$T_c = \sqrt{2\,T_s\,T_f} \qquad (1)$$

Young의 연구는 일반적인 시스템에서의 오류 발생률에 따른 최적의 체크포인팅 간격을 산출하였다. 이후 태스크의 종류에 따라 실시간성이나 데드라인 등을 고려하여 각 시스템에 최적 화 된 체크포인트 주기에 관한 여러 연구[1]가 진행되어 왔으나 대부분의 연구가 고정 주기의 체크포인트 방식을 따른다. 고정 된 체크포인트 주기는 관리자에 의해 결정되기 때문에 주기 계산을 위한 오버헤드가 적지만 시스템의 현재 상태를 반영하지 못하기 때문에 불필요한 체크포인트를 수행하거나 필요한 시기에 체크포인트를 하지 못하는 경우가 발생한다.

Ⅲ. 메트릭 값에 따른 동적 체크포인트 주기 산출

그림 1은 제안하는 동적 체크포인트 알고리즘의 개념을 보이고 있다. 일반적인 장애 관리 시스템에서 관리자는 메트릭 값을 일정 주기로 수집하여 해당 값이 경계값을 초과 할 경우 자동적으로 회복 되도록 조치한다. 이러한 경우 메트릭 값들은 대체로 선형적으로 증가하는 추세를 보인다. 따라서 각 메트릭 수집 시 이 전 값과 현재 값의 기울기를 통해 현재 메트릭 값의 추세를 예측하여 경계값을 초과하는 시점을 예상할 수 있다. 그림1에서 보이는 PTA(Predicted Time to Action)는 예측 시점으로부터 회복 액션이 수행 될 시점까지의 시간차를 뜻한다. 제안하는 알고리즘은 PTA 구간에 고정 횟수의 체크포인트를 삽입해 체크포인트 주기를 계산한다.

자원 모니터링은 짧은 주기로 수행되기 때문에 각 값은 순간적인 변동이 심할 수 있다. 따라서 각 메트릭들은 식(2)에 따른 메트릭 개수(N)의 값들을 산술평균하여 사용한다. P_c는 모니터링 주기, C_s는 체크포인트의 수행 시간을 뜻한다.

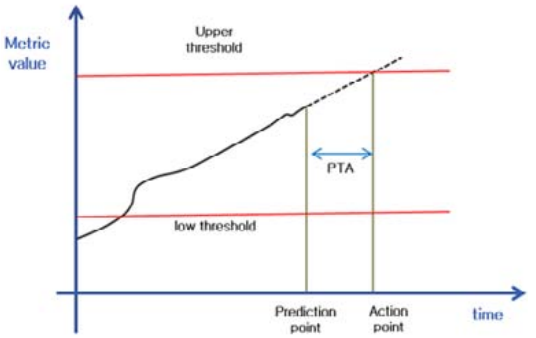

그림 1. 제안하는 체크포인트 알고리즘
Fig. 1. Proposing checkpoint algorithm

$$\mathrm{N} = \mathrm{MAX}\left(\frac{C_s}{P_c}, 2\right) \qquad (2)$$

체크포인트 주기가 체크포인트 동작의 소요 시간보다 빠를 경우, 하나의 자원에 대한 체크포인트 동작이 중복적으로 일어 날 가능성이 있기 때문에 위 식을 통해 체크포인트 주기 계산 간격이 수행 간격보다 길도록 보장 한다.

PTA는 식(3)과(4)를 통해 도출 된다. 식(3)에서 V_{curr}과 V_{pre}는 가장 최근과 바로 이전 메트릭들의 산술평균값이며 V_{th}는 회복 액션을 수행하여야 하는 메트릭의 경계값을 나타낸다. PTA는 경계값 까지의 소요 시간을 뜻하므로 두 산술평균 된 메트릭 사이의 기울기를 통해 식(4)로 도출 된다.

$$V_{curr} + \frac{V_{curr} - V_{pre}}{N^* P_c} * PTA = V_{th} \qquad (3)$$

$$PTA = \frac{V_{th} - V_{curr}}{V_{curr} - V_{pre}} * N^* P_c \qquad (4)$$

메트릭 값의 변동이 없는 경우에는 분모가 0이 될 수 있으므로 체크포인트 주기는 시스템의 장애 주기를 활용한 고정적인 값을 쓰거나 별도로 설정한 최대 PTA값을 활용한다.

PTA가 이전 계산주기에 산출 된 값보다 매우 짧게 계산되었지만 이 전에 계산 된 주기로 체크포인트 타이머가 동작 중일 경우, 갑작스런 상황에 대처하지 못한 경우가 생길 수 있다. 이러한 경우에는 타이머를 즉시 리셋하여 체크포인트 주기를 재설정 할 필요가 있다. 타이머 리셋은 계산 된 PTA가 체크포인트 소요

시간의 2배보다 작은 경우에 수행한다. 이 조건은 PTA 사이에 최소한 1번의 체크포인트 저장을 보장하며 체크포인트 즉시 실행 후 타이머 리셋이 이루어진다.

Ⅳ. 시뮬레이션 결과

모의실험을 위해 리눅스 컨테이너를 사용하는 가상환경 상에서 제안하는 시스템을 구현하였으며 메모리 누수 코드를 의도적으로 동작시켜 장애 환경을 구성 후 체크포인트 주기를 관찰하였다. 모니터링 주기인 P_c는 1초로 지정하였고 초기 체크포인트 소요시간인 C_s는 평균적으로 컨테이너가 소모하는 5초로 측정하였다. 회복액션을 수행하는 경계값인 V_{th}는 90%로 하였고 PTA구간에 총 3번의 고정적인 체크포인트를 수행하도록 N은 3으로 설정하였다. 최대 체크포인트 주기는 30분으로, 이는 Young의 연구에 의하여 약 3일에 한번 장애가 발생 할 시 최대의 효율을 가지는 체크포인트 주기를 뜻한다. 그림 2는 메모리 누수 코드 동작 시 메모리와 CPU의 활용률을 보인다. 이 때, 제안하는 시스템은 그림 3과 같은 PTA와 체크포인트 주기를 보였다. 메트릭 값이 크게 변화함에 따라 PTA

그림 2. CPU / Memory 활용률
Fig. 2. CPU / Memory utilization

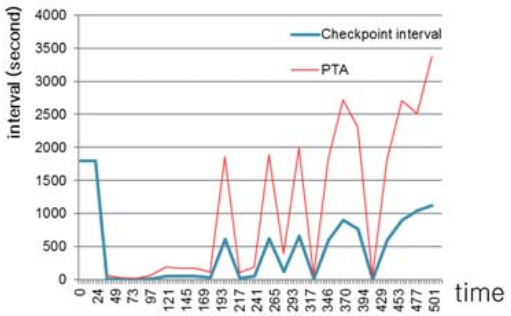

그림 3. 체크포인트 주기 변화
Fig. 3. Variation of checkpoint period

표 1. 체크포인트 방식에 따른 효율 비교
Table 1. Efficiency of checkpoint methods

method	total checkpoint	valid checkpoint
Young's algorithm	161	4
proposed algorithm	90	41

가 크게 감소하고 체크포인트 주기 역시 크게 감소하였다. 이 후 CPU 활용량이 90% 이상 넘어 갈 시, 컨테이너에 회복 조치가 수행되어 활용률은 급감하고 주기는 회복되는 동작을 보인다. 표1은 8시간의 장애 주기를 갖는 시스템에서 고정 체크포인트 방식과의 효율 차이를 보인다. 시뮬레이션은 하루 간 총 체크포인트 횟수와 자원 활용률이 증가하는 구간의 유효 체크포인트 횟수를 측정하였으며, 고정 체크포인트 방식은 Young의 방식을 활용하여 536초의 주기를 갖는 경우로 계산하였다. 제안하는 알고리즘은 평시 상태의 체크포인트 횟수는 고정 방식보다 적지만, 장애 구간에서 체크포인트 횟수가 급증하여 유효 체크포인트 횟수가 높게 측정되었다. 따라서 고정 체크포인트 방식에 비해 실시간 장애에 빠르게 대응하였고, 높은 효율을 보였다.

Ⅴ. 결 론

본 논문에서는 가상화 환경에서 모니터링을 통해 얻은 메트릭값의 상태에 따라 체크포인트 주기를 동적으로 변경하는 방법을 제안하였다. 메모리 및 CPU 활용률과 같은 메트릭 값들이 장애 상태로 평가되는 경계값에 근접 할수록 짧은 체크포인트 주기를 가진다. 이를 통해 관리자는 장애 발생 시 정상 상태로 회복시킬 확률을 높일 수 있다.

References

[1] K. S. Byun and J. H. Kim, "A study on optimal checkpointing interval in real-time systems" *J. KICS*, vol. 26, no. 7, pp. 1220-1226, Jul. 2001.

[2] John W. Young, "TA first order approximation to the optimum checkpoint interval" *ACM*, vol. 17, no. 9, Jan. 1974.