Software-Defined RAID 기반 장애복구 기법과 실증 테스트

(Verification Test of Failover Recovery Technique based on Software-Defined RAID)

차병래*, 최명수**, 박선*, 김종원*

(ByungRae Cha*, MyeongSoo Choi**, Sun Park*, JongWon Kim*)

요 약

본 논문은 분산 저장환경에 사용되는 RAID와 네트워크 가상화 기술을 융합하며 Software Defined Storaging 방법을 제안한다. 제안방법은 저장장치를 소프트웨어 기반으로 설계함으로써 유연한 제어 및 관리를 제공하여 물리적인 스토리지의 용량과 재해 복구비용을 절감할 수 있다. 제안된 Software-Defined RAID 기반의 장애복구 기법을 실질적인 퍼블릭 클라우드인 AWS와 구글 스토리지를 이용하여 실증 테스트 및 성능을 비교한다.

■ 중심어 : 분산 스토리지; 소프트웨어-정의 스토리지; RAID; 네트워크 가상화; 소프트웨어-정의 인프라스트럭처

Abstract

This paper proposes a software defined storaging method to converge the network virtualization technique and the RAID of distributed storage environment. The proposed method designs software based storage which it apply a flexible control and maintenance of storages. In addition, the method overcomes the restricted of physical storage capacity and cut costs of data recovery. The proposed failover recovery technique based on Software-Defined RAID has been tested the substantial verification and the performance using public AWS and Google Storage.

■ keywords : Distributed Storage; Software-Defined Storage; RAID; Network Virtualization; SDI

Ⅰ. 서 론

오늘날 휴대용 디지털 기기의 광범위한 보급과 하드웨어 및 소프트웨어의 발달로 인하여 텍스트를 비롯한 이미지 및 비디오 영상 등의 다양한 빅데이터가 생성 및 이러한 데이터의 획득과 저장 그리고 재사용은 다양한 분야에서 매우 일반화 되고 있으며, 특히, IoT 환경에서의 사용자들은 보다 손쉬운 방법으로 다양한 정보를 원하고 있는 상황이다. Software-Defined RAID Framework는 데이터 저장을 위한 물리적 저장장치의 문제 발생 시에 논리적으로 퍼블릭 클라우드의 스토리지 또는 다양한 외부의 저장 공간에 임시적으로 데이터를 저장하며, 물리적 저장장치가 복구가 완료되면 역으로 외부의 저장 공간으로부터 물리적 저장장치로의 데이터를 복구하는 기능을 제공하고자 한다. 본 연구의 Software-Defined RAID 기반 장애복구 기법을 통해 퍼블릭 클라우드 또는 외부 저장 공간에 임시적인 데이터 백업 및 복구 기능의 운영이 가능하다.

컴퓨팅과 네트워킹, 그리고 스토리지의 패러다임 변화에 따라 클라우드 인프라 및 데이터센터의 저장장치에 사용되는 RAID 기술[1]과 네트워크 가상화 기술[2]을 융합하며, 저장장치를 소프트웨어 기반으로 구현함으로써 유연한 제어 및 관리를 제공하여 물리적인 스토리지의 용량과 재해복구 비용을 절감할 수 있는 Software Defined Storaging 방법[3]을 제안하고 이를 적용한 프로토타입 모듈을 설계 및 개발한다. 현재 데이터 센터의 저장장치는 기계적인 수명과 환경적인 요인으로 인해 장애가 발생하여 중요한 데이터가 손실되고 있다. 장애극복을 위하여 센터의 저장장치들은 대부분 하드웨어 RAID로 구성되어 있으나 디스크 드라이브의 개수가 늘어날수록 디스크 고장의 확률도 증가한다. 데이터센터의 데이터 보호를 위하여 사용되는 스토리지의 안정성/확장성/유연성을 확보하고 물리적 사용량을 절감하기 위해, 네트워킹과 RAID 기술을 이용한 소프트웨어 기반의 제어 가능한 가상화 스토리지 기술의 개발이 필요하다.

* 정회원, 광주과학기술원 정보통신공학부

** 정회원, 제노테크(주)

이 논문은 2015년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임(2015H1C1A1035823).

접수일자 : 2016년 01월 26일 게재확정일 : 2016년 03월 03일

수정일자 : 2016년 02월 22일

교신저자 : 김종원, e-mail : jongwon@smartx.kr

각기 다른 지역에 분산되어 있는 클라우드 데이터 센터 내에서 물리적으로 분산된 저장 매체는 각각의 개별적인 스토리지로 인식된다. 이러한 지역적 제약사항을 극복하여 분산된 개별 저장매체를 연결하여 하나의 저장매체로 인식하려면 저장매체를 논리적으로 추상화하여 논리적으로 관리하는 기술이 필요하다. 제안하는 Software-Defined RAID 기반 장애복구 모듈은 물리적 디스크와 로컬 기반의 Hot Spare/RAID 드라이브를 분산 환경에서 사용하는 분산 Virtual Hot Spare/RAID 및 장애복구를 지원한다. 제안 및 개발된 프로토타입의 모듈은 클라우드 내의 서버에 자료를 저장하는 Software-Defined RAID와 RAID의 장애복구 및 관리를 위한 Software-Defined Hot Spare Server로 구분된다. 일반적으로 모듈을 개발하여 클라우드 환경에 사후에 적용시킬 경우 다양한 문제가 발생할 수 있으며, 개발 모듈을 클라우드 환경에 적합하도록 조율할 때 많은 시간과 노력이 소요된다. 본 논문은 분산 저장환경에 사용되는 RAID와 네트워크 가상화 기술을 융합하며 Software Defined Storaging 방법을 제안 및 가용성을 확인한다. 제안방법은 저장장치를 소프트웨어 기반으로 설계함으로써 유연한 제어 및 관리를 제공하여 물리적인 스토리지의 용량과 재해 복구비용을 절감할 수 있다.

Ⅱ. 관련 연구

1. SDx: 소프트웨어-정의 패러다임의 확산

SDx (Software-Defined Anything/Everything)란 ICT 인프라의 하드웨어와 소프트웨어를 분리해 프로그래밍을 통한 인프라 관리를 강화하고 상호운용성을 높이는 표준적인 방법론을 만들려고 하는 일련의 포괄적인 움직임(즉 패러다임)을 지칭하는 용어다. 현재 이러한 미래지향적인 패러다임을 실현하기 위한 활동들이 활발하게 전개되고 있으며, 오픈스택(OpenStack), 오픈플로우(OpenFlow), 오픈 컴퓨트 프로젝트(Open Compute Project: OCP) 등이 대표적이다. 또한 시스코, EMC, HP, IBM, 인텔, 마이크로소프트, 화웨이, 넷앱, 시만텍, VM웨어 등이 SDx 패러다임에 관심을 표명한 주요 업체들이다. SDx 중에서 적용 측면에서 가장 먼저 시작한 것은 SDN 분야이다. 개별 네트워킹 장비를 로직 부분과 정책 부분으로 나눠 이를 소프트웨어로 구현해 통합적으로 관제(monitor & control)하려는 시도들이다. ICT 수요자들이 신규 장비 도입을 줄이고 고가의 장비 대신 범용 장비를 도입해 활용할 수 있어 기존 시장 질서를 뒤엎을 수 있는 파괴적인 기술로 가트너는 평가하고 있다. Gartner의 분석가인 필립 다우슨은 "개별 SDx 영역의 성숙도를 보면 초기부터 고도화된 단계까지 천차만별이나, 전반적인 SDx 수준을 보면 아직 촉발기 단계로 평가된다"고 말했다. 이어 "SDx는 결국 특정 업체에 대한 종속에서 벗어나 수요자가 더 자유롭게 제품을 선택할 수 있으므로, SDx의 진정한 잠재력을 구현하려면 애플리케이션과 소프트웨어 영역으로 더 공격적으로 확장해야 한다"라고 덧붙였다.

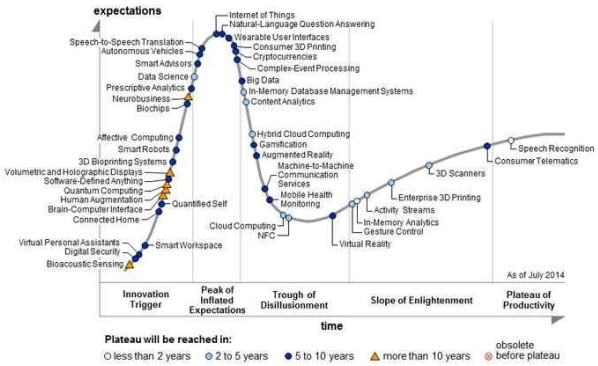

그림 1. 2014 Gartner Hype Cycle Curve

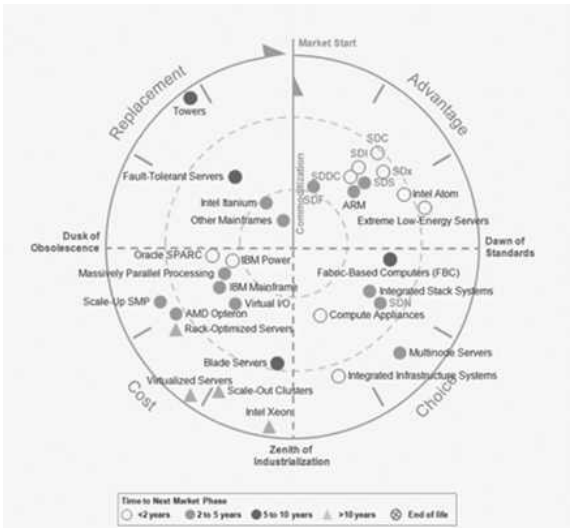

그림 2. 2014 Gartner Market Cycle

Gartner에서 발행한 '2014 Gartner Hype Cycle Curve'에서 SDx와 관련된 부분을 보면 [그림 1]과 같다. 가상화를 시작으로 SDN에 힘입어 화두가 된 SDx 개념은 이제 소프트웨어-정의 스토리지 및 소프트웨어-정의 데이터센터가 얼마나 빠르게 확산하느냐에 따라 성장이 좌우되고 있다. SDN은 이미 Google, Microsoft 등 주요 소프트웨어 회사에 의해 가치가 입증되었으며, 현재는 기술의 안정화 단계에 접어들고 있다. 시장조사기관 IDC에 따르면 전 세계 SDN 시장은 향후 5년간 연평균 89.4%의 성장세로, 2014년 9억 6천만 달러 규모에서 2018년 80억 달러 규모로 급성장할 것으로 전망하고 있다. 또한 [그림 2]의 Gartner의 서버 기술 및 SDx 관련 Market Cycle을

살펴보면 서버 기술들이 소프트웨어-정의 데이터 센터의 장점을 활용하는 방향으로 발전하는 추세를 보이고 있다.

2. Software-Defined RAID 기반 장애복구 기법

기존의 데이터센터의 기반이 되는 저장소는 주로 하드웨어 RAID로 구성되어 있다. 이들 저장소의 재해복구 및 유지관리를 위하여 추가적인 하드웨어 장치가 필요하며 저장소가 늘어날수록 이들을 위한 추가적인 비용이 계속적으로 증가하고 있다. 이러한 이유로 데이터센터의 저장소에 대한 효율적이면서 저비용의 관리기술이 필요하다.

본 논문은 이를 위한 데이터센터와 클라우드의 저장소를 네트워크 가상화로 추상화하여 하나의 저장소(unified storage)로 유지 및 관리할 수 있는 Software Defined Storaging 구조를 설계하였다. 다음 그림은 본 논문에서 제안한 Software Defined Storaging 기술의 개념도와 흐름도는 [그림 3]과 [그림 4]에 나타내며 세부 구성기술은 다음과 같다.

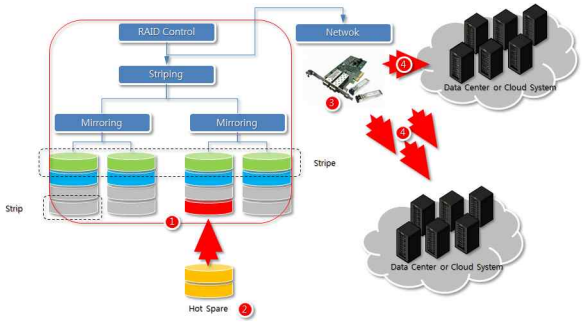

그림 3. Software Defined Storaging 기술의 개념도

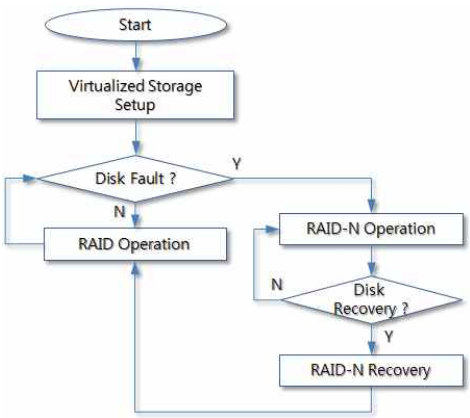

그림 4. Software Defined Storaging 기술의 FlowChart

가. RAID 기술: 스트라이핑과 미러링, 패리티 같은 RAID 기술은 여러 RAID 레벨을 정의하는데 기초가 되며, 이 기술은 RAID 집합의 데이터 가용성과 성능을 결정한다.

나. Hot Spare 기술: Hot Spare는 고장 난 디스크를 잠시 대체하기 위하여 RAID Array에 있는 여분의 드라이브를 의미한다[4].

다. BYOE & Secret Sharing Schemes: BYOE은 (Bring Your Own Encryption) 클라우드 서비스의 고객이 자신의 암호화 소프트웨어를 사용하여 자신의 암호화 키를 관리 할 수 있는 클라우드 컴퓨팅 보안 모델이다[5]. Secrete Sharing Scheme (SSS)는 비밀분산법(시스템)으로 어떠한 비밀정보를 몇 개의 분산정보로 부호화하고, 그 분산정보가 어느 일정개수 이상 모이면 원래의 비밀정보가 복원되지만, 그것보다 작은 분산정보에서는 원래의 비밀 정보를 전혀 알 수 없다는 부호화법이다[6].

라. Mutipath-TCP 기술: 다중 경로 TCP는 네트워크 전반적인 트래픽 엔지니어링 효과를 얻을 수 있는 장점이 있으며, VoIP, IPTV, 게임 등과 같은 요구가 많은 서비스에게 신뢰성(reliability)을 제공한다[7].

III. 저장매체 추상화에 의한 Software-Defined RAID 기반 장애복구 기법

본 논문은 논리적으로 한 저장매체로 추상화하는 Software-Defined RAID 기반 장애복구를 위한 Software-Defined RAID Framework의 설계, 구현 및 실제 활용 방법에 대해 상세히 기술한다. Software-Defined RAID Framework는 물리적 저장장치의 문제 발생 시에 논리적으로 퍼블릭 클라우드의 스토리지 또는 다양한 외부의 저장 공간에 임시적으로 데이터를 저장하며, 물리적 저장장치가 복구가 완료되면 역으로 외부의 저장 공간으로부터 물리적 저장장치로의 데이터를 복구하는 기능을 제공한다. 본 Software-Defined RAID Framework를 통해 퍼블릭 클라우드 또는 외부 저장 공간에 임시적인 데이터 백업 및 복구 기능의 운영이 가능하다.

각기 다른 지역에 분산되어 있는 클라우드 데이터 센터 내에서 물리적으로 분산된 저장 매체는 각각의 개별적인 스토리지로 인식된다. 이러한 지역적 제약사항을 극복하여 분산된 개별 저장매체를 연결하여 하나의 저장매체로 인식하려면 저장매체를 논리적으로 추상화하여 논리적으로 관리하는 기술이 필요하다. 개발되는 Software-Defined RAID 기반 장애복구 모듈은 물리적 디스크와 로칼 기반의 Hot Spare/RAID 드라이브를 분산 환경에서 사용하는 분산 Virtual Hot Spare/RAID 및 장애

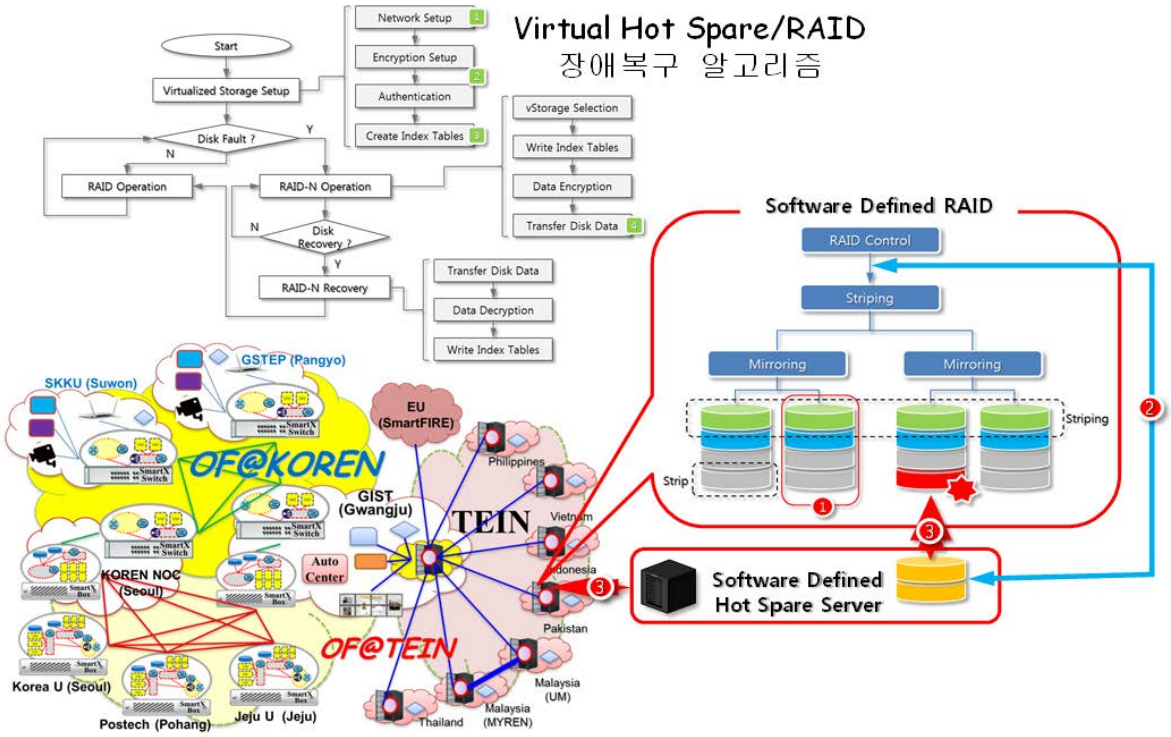

그림 5. Software-Defined RAID기반 장애복구 모듈.

복구를 지원한다. 이를 위해서 자동설정 모듈에서 Chef DevOps 도구를 활용하여 자동으로 설치되는 OpenStack[8]과 Ceph[9]를 기반으로 Software-Defined RAID 기반 장애복구 모듈의 개발이 필요하다. 개발되는 모듈은 클라우드 내의 서버에 자료를 저장하는 Software-Defined RAID와 RAID의 장애복구 및 관리를 위한 Software-Defined Hot Spare Server로 구분한다. 일반적으로 모듈을 개발하여 클라우드 환경에 사후에 적용시킬 경우 다양한 문제가 발생할 수 있으며, 개발 모듈을 클라우드 환경에 적합하도록 조율할 때 많은 시간과 노력이 소요된다. 본 과제에서는 OF@TEIN/OF@KOREN 테스트베드 인프라를 활용하여 모듈을 직접 개발함으로써 클라우드에 적용 시에 발생하는 문제들을 최소화시킨다.

[그림 5]는 OF@TEIN/OF@KOREN 인프라를 활용하여 개발예정인 Software-Defined RAID 기반의 장애복구 모듈의 구성을 보여준다. 개발 모듈은 다음과 같은 시나리오로 운영된다. 예를 들어 KOREN NOC (NIA) 사이트의 서버에 저장소를 위해서 Software-Defined RAID가 위치하며, 저장소 관리 및 장애복구 지원을 위해서 Software-Defined Hot Spare Server가 OF@TEIN/OF@KOREN 인프라에 접속된다. [그림 5]의 ①은 NOC의 서버가 자료를 저장 시 RAID에 미러링되어 저장되며, 동시에 [그림 5]의 ②와 같이 Software-Defined Hot Spare Server 자료들을 저장한다. 만약 서버에 장애가 발생하면, [그림 5]의 ③과 같이 Software-Defined Hot Spare Server로부터 자료를 복구하면서 동시에 NOC서버의 저장 자료들이 [그림 5]의 ②와 같이 Hot Spare Server에 저장됨으로써 사용자를 저장소의 장애사실을 전혀 인식하지 못하도록 한다.

또한 Virtual Hot Spare/RAID 관리 및 장애복구 모듈은 크게 3 단계로 구성된다. 첫 번째 단계로는 가상된 스토리지의 설정 과정이며, 디스크 결함에 발생하기 전에 사전대응하기 위하여 다수의 스토리지 사이트에 대한 기본 설정 작업을 한다. 두 번째는 디스크 결함을 체크하는 과정으로, 주기적으로 디스크의 결함을 체크하여 결함이 발견되지 않으면 RAID의 데이터 조작 연산을 수행한다. 만약 디스크 결함이 발생하게 되면 RAID-N의 데이터 조작 연산을 수행한다. 마지막으로 RAID-N의 데이터 조작 연산이 수행 중에 디스크 결함이 복구되면, RAID-N 복구 절차를 수행한다. RAID-N 복구 절차가 수행이 완료되면 RAID의 데이터 조작 연산을 다시 수행하여 스토리지가 정상적으로 동작한다. 가상화된 스토리지 설정은 먼저 다수의 스토리지 사이트에 대한 네트워크 설정과 보안을 위한 암호화와 인증, 그리고 RAID-N의 데이터 조작 연산의 수행 결과의 위치를 기록할 색인을 생성한다. RAID-N의 데이터 조작 연산은 디스크 결함에 의한 데이터를 저장할 가상화된 스토리지를 선택하고, 선택된 가상화된 스토리지에 기록할 데이터의 색인을 기록하며, 데이터 전송 전에 데이터를 암호화하여 가상화된 스토리지로 전송한다. RAID-N 복구절차는 색인 정보에 의해서 각각의 가

상화된 스토리지에 전송된 데이터들을 수신하고 복호화 과정을 진행한다. 수신된 데이터의 무결성이 확인되면 색인에서 기록을 삭제한다.

논리적으로 한 저장매체로 추상화하는 Software-Defined RAID 기반 장애복구 모듈은 물리적 디스크와 로칼 기반의 Hot Spare/RAID 드라이브를 분산 환경에서 사용하는 분산 Virtual Hot Spare/RAID 지원 및 장애복구 모듈로 구성된다. 클라우드 자원 인프라 자동화에 연계한 스토리지 자원 자동화에 의하여 Self-managed 스토리지를 빠르게 제공하여 비즈니스에서 요구하는 업무 연속성을 지원하는 새로운 시장의 창출 가능하며, Software-Defined Storage 패러다임의 확대로 인하여 가상 인프라에서 스토리지 자원을 사용하는 트렌드가 일반화됨에 따라 기존 스토리지 구매, 사용 방법에 변화가 예상되며, 이에 부합하여 신 시장을 선제적으로 대응하는 전략의 수립이 가능하다.

1. SD RAID 기반 장애복구 모듈의 설계

분산 저장 환경의 데이터 공유 및 관리를 위한 소프트웨어-정의 저장 시스템은 논리적으로 한 저장매체로 추상화하는 Software-Defined RAID 기반 장애복구 모듈을 설계하며, 물리적 디스크와 로컬 기반의 Hot Spare/RAID 드라이브를 분산 환경에서 사용하는 분산 Virtual Hot Spare/RAID 지원 및 장애복구 모듈을 설계 및 개발한다.

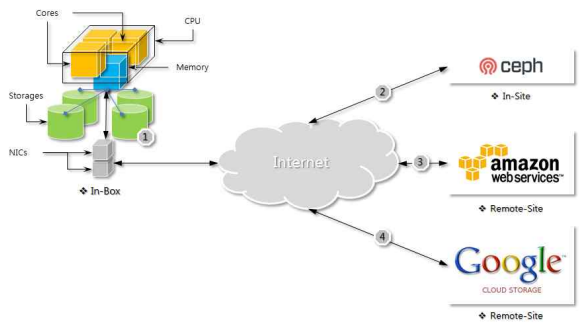

그림 6. Software-Defined RAID 기반 장애복구 모듈의 설계를 위한 시스템 환경.

Software-Defined RAID 기반 장애복구 모듈의 설계를 위한 시스템 환경은 [그림 6]과 같이 나타낼 수 있으며, Software-Defined RAID 기반 장애복구 모듈의 FlowChart는 [그림 7]에 나타낸다. 데이터를 백업할 영역을 크게 3 가지로 In-Box([그림 6]의 ① 참조), In-Site([그림 6]의 ② 참조), 그리고 Remote-Site([그림 6]의 ③와 [그림 6]의 ④ 참조)로 구분하며, 본 논문에서는 주로 Remote-Site인 AWS 와 Google Storage에 초점을 맞춰 SD-HotSpot RAID 기반 장애복구 모듈들의 프로토타입 설계 및 개발을 수행하였으며, SD-HotSpot 프레임워크를 정의한다.

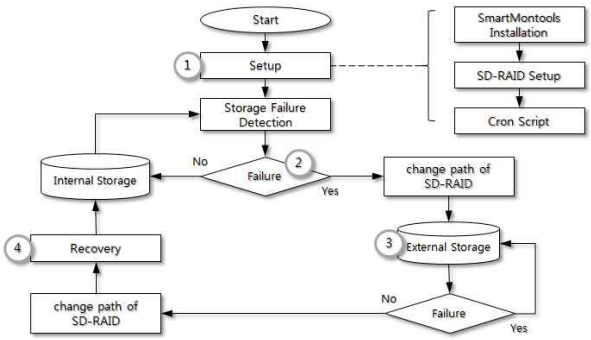

그림 7. Software-Defined RAID 기반 장애복구 모듈의 FlowChart.

Software-Defined RAID 기반 장애복구 모듈의 FlowChart는 크게 4 가지 기능으로 구분되며, SD RAID 기반 장애복구를 위한 사전 설정 작업([그림 7]의 ① 참조), 스토지 고장 검출([그림 7]의 ② 참조), 외부 스토리지 백업([그림 7]의 ③ 참조), 그리고 외부 스토리지 데이터의 내부 물리적 저장장치로의 복구([그림 7]의 ④ 참조)로 구성된다.

2. 클라우드 스토리지 의 SD RAID 기반 장애복구 모듈의 설계

가. AWS 의 SD RAID 기반 장애복구 모듈의 설계

Remote-Site AWS(Amazon Web Service)를 이용한 SD RAID 기반 장애복구 모듈의 설계를 위한 시스템 환경을 [그림 8]과 같이 나타내며, Remote-Site AWS를 이용한 SD RAID 기반 장애복구 모듈의 FlowChart는 [그림 9]와 같이 나타낸다.

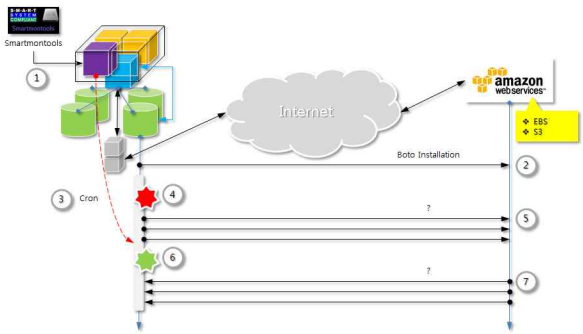

그림 8. Remote-Site AWS를 이용한 SD RAID 기반 장애복구 모듈의 시스템 환경.

Remote-Site AWS(Amazon Web Service)를 이용한 SD RAID 기반 장애복구 모듈은 [그림 8]의 시스템 환경과 [그림 9]의 FlowChart 에 나타낸 것과 같은 절차로 진행되며, 먼저, In-Box 안에 Smartmontools API[10]가 설치되고 물리적 저장 장치를 모니터링하게 되며, Python의 boto API[11]를 설치하는 과정으로 SD RAID 기반 장애복구 모듈은 개발이 진행된다. boto API의 설치가 완료되면, 더불어 Paramiko 패키지와 Ec2ool API의 설치가 진행된다.

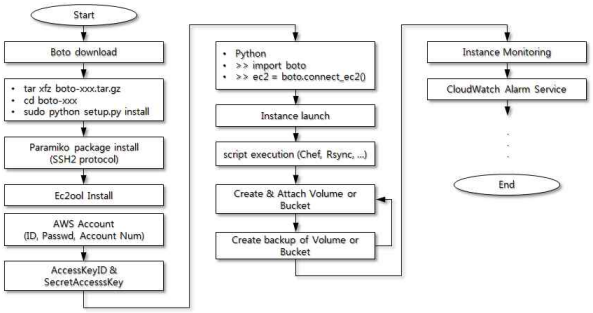

그림 9. Remote-Site AWS를 이용한 SD RAID 기반 장애복구 모듈의 FlowChart.

기본적인 패키지와 API가 설치가 완료되면, AWS의 계정을 생성함과 동시에 AccessKeyID와 SecretAccessKey를 획득하게 된다. 이어서, boto 모듈을 로딩과 더불어 인스턴스를 시작하고 Chef, RSync 등의 필요한 다양한 관리 패키지를 설치하는 스크립트를 실행하게 된다. AWS의 버켓 또는 볼륨을 생성과 더불어 인스턴스에 부착하게 되며, 버켓 또는 볼륨을 백업용도로 초기화를 하게 되며, 이로써 SD RAID 기반 장애복구 모듈의 백업 공간이 확보된다. 또한 SD RAID 기반 장애복구 모듈의 백업 공간에 대한 모니터링과 CloudWatch Alarm 서비스 등에 의한 관리가 진행된다.

나. Google의 SD RAID 기반 장애복구 모듈의 설계

Remote-Site Google Storage를 이용한 SD RAID 기반 장애복구 모듈의 설계를 위한 시스템 환경을 [그림 10]과 같이 나타내며, Remote-Site Google Storage 를 이용한 SD RAID 기반 장애복구 모듈의 FlowChart는 [그림 11]과 같이 나타낸다.

Remote-Site Google Storage를 이용한 SD RAID 기반 장애복구 모듈은 [그림 10]의 시스템 환경과 [그림 11]의 FlowChart 에 나타낸 것과 같은 절차로 진행되며, 먼저, In-Box 안에 Smartmontools API가 설치되고 물리적 저장 장치를 모니터링하게 되며, Python의 boto API를 설치하는 과정으로 SD RAID 기반 장애복구 모듈은 개발이 진행된다. boto API의 설치가 완료되면, 더불어 Paramiko 패키지와 SSH2 프로토콜의 설치가 진행된다.

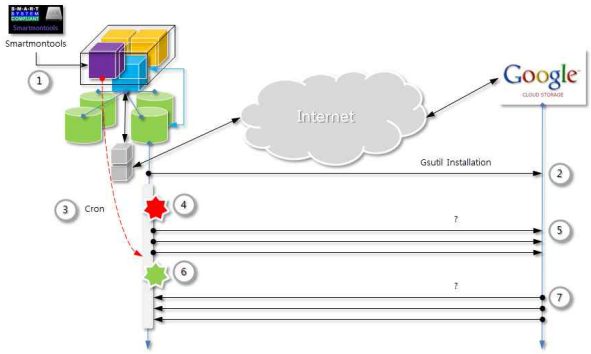

그림 10. Google Storage를 이용한 SD RAID 기반 장애복구 모듈의 시스템 환경.

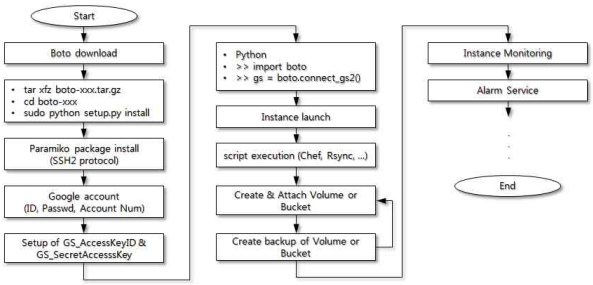

그림 11. Google Storage를 이용한 SD RAID 기반 장애복구 모듈의 FlowChart.

기본적인 패키지와 API, 그리고 프로토콜의 설치가 완료되면, Google의 계정을 생성함과 동시에 GS_AccessKeyID와 GS_SecretAccessKey를 획득하게 된다. 이어서, boto 모듈을 로딩과 더불어 인스턴스를 시작하고 Chef, RSync 등의 필요한 다양한 관리 패키지를 설치하는 스크립트를 실행하게 된다. Google Storage의 버켓 또는 볼륨을 생성과 더불어 인스턴스에 부착하게 되며, 버켓 또는 볼륨을 백업용도로 초기화를 하게 되며, 이로써 SD RAID 기반 장애복구 모듈의 백업 공간이 확보된다. 또한 SD RAID 기반 장애복구 모듈의 백업 공간에 대한 모니터링과 Alarm 서비스 등에 의한 관리가 진행된다.

Ⅳ. Software-Defined RAID 기반 장애복구의 실증 테스트

In-Site의 물리적 저장 장치의 문제가 발생하게 되면 Remote-Site의 AWS와 Google Storage를 boto API를 이용하여 SD RAID 기반 장애복구가 가능한 모듈을 구현 및 테스

트를 진행한다.

1. SD RAID 기반 장애복구 모듈의 환경 및 설정

SD RAID 기반 장애복구 모듈의 테스트를 위한 환경으로 Server와 VM의 스펙은 다음의 [표 1]과 [표 2]에 나타낸다. Boto와 AWS는 접근키 ID와 비밀 접근키를 통해서 인증 후에 원하는 명령어를 전송해 결과를 출력하게 되며 [그림 12]에 나타낸다.

표 1. 스토리지 테스트를 위한 Server 스펙

CPU	Intel® Xeon® cpu x5670@ 2.93GHz * 2
MEMORY	32768MB
NETWORK	1G NetworkCard
HDD	320G * 2

표 2. 스토리지 테스트를 위한 VM 스펙

Memory	1024MB
Storage	16GB+100GB
Network	1개
OS	Ubuntu

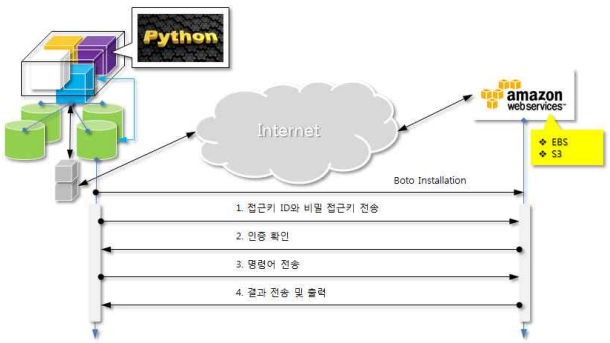

그림 12. boto를 이용한 AWS의 접근방식.

2. AWS & Google Storage의 테스트

클라우드 스토리지 AWS와 Google Storage의 버킷을 통한 데이터 크기별로 1MB, 10MB, 100MB, 1G, 2G, 5G의 파일 업로드 및 다운로드 테스트를 실시한다.

클라우드 스토리지 AWS 버킷을 통한 데이터 크기별로 파일 업로드 및 다운로드 테스트를 실시한 결과를 [표 3]과 [표 4]에 나타낸다.

표 3. AWS의 데이터 업로드 테스트 결과

횟수	Data Size					
	1M	10M	100M	1G	2G	5G
1	1.51	3.68	12.13	105.02	355.53	816.23
2	1.51	3.55	12.21	100.55	201.91	496.71
3	1.93	4.37	13.3	99.95	199.9	498.82
4	1.96	3.86	12.09	99.54	536.98	536.36
5	2.12	3.48	12.59	257.57	215.07	526.06
6	2.29	3.65	17.4	600.25	237.7	606
7	1.89	6.53	12.45	163.25	199.82	1397.49
8	1.74	3.32	12.05	377.45	200.67	536.92
9	1.64	3.92	12.69	103.69	617.64	494.72
10	1.99	3.34	47.5	104.43	198.07	490.15
평균	1.858	3.97	16.441	201.17	296.329	639.946

표 4. AWS의 데이터 다운로드 테스트 결과

횟수	Data Size					
	1M	10M	100M	1G	2G	5G
1	1.88	3.6	15.78	155.49	1206.36	774.34
2	1.89	3.59	87.17	144.62	316.7	839.73
3	4.13	4.29	15.56	277.67	301.57	1100.68
4	1.9	4	19.78	134.18	407.47	728.18
5	2.09	3.68	19.24	152.65	300.7	735.41
6	2.5	3.62	22.37	397.38	576.1	949.95
7	1.88	3.87	17.82	168.96	342.33	689.19
8	1.68	3.57	20.71	188.31	310.37	789.24
9	1.73	4.27	20.69	245.37	374.28	923.28
10	1.79	8.21	16.5	237.35	427.76	963.16
평균	2.147	4.27	25.562	210.198	456.364	849.316

표 5. Google Storage의 데이터 업로드 테스트 결과

횟수	Data Size					
	1M	10M	100M	1G	2G	5G
1	3.72	5.99	12.1	98.11	194.73	479.89
2	7.56	4.37	12.05	96.52	196.72	481.85
3	4.19	5.82	10.3	96.73	195.01	481.89
4	6.77	5.07	13.33	97.08	192.55	480.41
5	3.45	3.26	11.65	97.13	193.66	481.62
6	2.8	3.05	15.59	96.68	194.28	483.52
7	5.87	4.49	12.36	96.71	194.42	486.11
8	4.06	3.13	10.83	98.26	192.71	483.79
9	5.71	2.22	12.75	97.18	194.04	483.54
10	2.45	4.09	11.43	98.31	192.43	483.39
평균	4.658	4.149	12.239	97.271	194.055	482.601

표 6. Google Storage의 데이터 다운로드 테스트 결과

횟수	Data Size					
	1M	10M	100M	1G	2G	5G
1	2.67	2.41	12.19	93.29	184.94	488.82
2	2.54	3.33	12.16	92.85	187.45	459.17
3	2	3.97	12.06	93.24	185.49	459.8
4	2.28	3.92	12.5	100.53	184.43	467.28
5	1.37	3.92	11.4	95.68	185.52	461.4
6	2.15	3.96	11.54	95.05	184.59	521.15
7	2.55	4.95	12.03	100.02	184.38	459.4
8	2.34	4.75	10.55	94.06	184.44	463.17
9	3	1.85	10.27	94.81	194.51	461.01
10	20.8	3.93	10.29	92.82	186.1	459.12
평균	4.17	3.699	11.499	95.235	186.185	470.032

클라우드 스토리지 Google Storage 버킷을 통한 데이터 크기별로 파일 업로드 및 다운로드 테스트를 실시한 결과를 [표 5]와 [표 6]에 나타낸다.

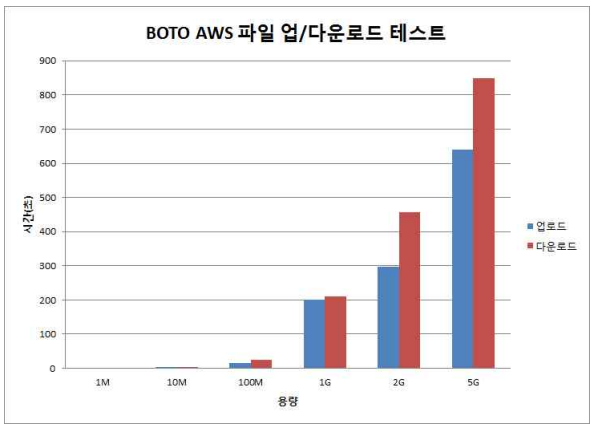

그림 13. AWS의 테스트 결과 그래프.

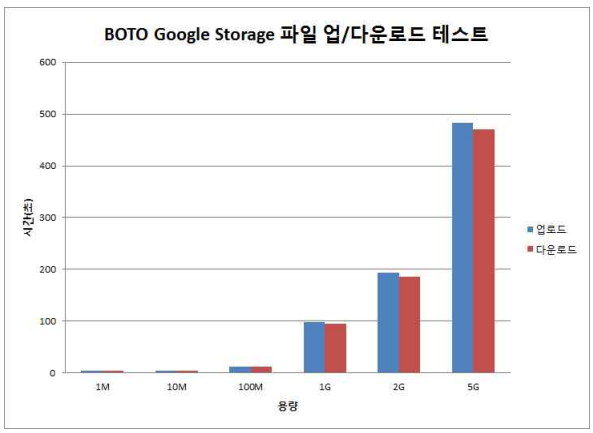

그림 14. Google Storage의 테스트 결과 그래프.

AWS와 Google Storage의 버킷을 통한 데이터 크기별로 파일 업로드 및 다운로드한 테스트 결과를 분석한 챠트를 [그림 13]과 [그림 14]에 나타낸다.

AWS와 Google Storage의 버킷을 통한 데이터의 업로드 테스트 결과와 다운로드 테스트 결과는 AWS 보다는 Google Storage가 더 빠름([그림 13]과 [그림 14]를 비교 참조)을 보여주고 있다. AWS의 버킷을 통한 데이터의 업로드와 다운로드 테스트 결과는 다운로드 속도보다는 업로드의 속도가 상당히 빠름을 나타내고 있다([그림 13] 참조). Google Storage의 버킷을 통한 데이터의 업로드와 다운로드 테스트 결과는 업로드 속도가 약간 더 빠르지만, 전체적으로 업로드와 다운로드 속도가 거의 동등함을 나타내고 있다([그림 14] 참조).

V. 결 론

본 논문은 분산 저장환경에 사용되는 RAID와 네트워크 가상화 기술을 융합하며 Software Defined Storaging 방법을 제안 및 설계하였다. 제안된 기술은 물리적 저장장치의 장애 문제를 소프트웨어 기반으로 저장 공간 대체 및 장배 복구 전략을 설계함으로써 유연한 제어 및 관리를 제공함과 더불어 물리적인 스토리지의 용량과 재해 복구비용을 절감할 수 있다. 또한 퍼블릭 클라우드 AWS와 Google의 클라우드 스토리지를 이용한 프로토타입을 개발하였으며, 이를 이용한 테스트를 수행하여 가능성을 검증 및 성능을 비교분석 하였다.

References

[1] RAID, http://en.wikipedia.org/wiki/RAID, 2016, 2.
[2] Network virtualizaiton, http://en.wikipedia.org/wiki/Network_virtualization, 2016, 2.
[3] Software-defined Storage, http://en.wikipedia.org/wiki/Software-defined_storage, 2016, 2.
[4] Hot spare, http://en.wikipedia.org/wiki/Hot_spare, 2016, 2.
[5] BYOE, http://whatis.techtarget.com/definition/BYOE-bring-your-own-encryption, 2016, 2.
[6] Secret Sharing, http://en.wikipedia.org/wiki/Secret_sharing, 2016, 2.
[7] Multiple-TCP, https://en.wikipedia.org/wiki/Multipath_TCP, 2016, 2.
[8] OpenStack, https://www.openstack.org/, 2016. 2
[9] Ceph, https://en.wikipedia.org/wiki/Ceph_(software), 2016, 2.
[10] Smartmontools API, https://www.smartmontools.org/, 2016, 2.
[11] boto API, https://aws.amazon.com/ko/sdk-for-python/, 2016, 2.

저 자 소 개

차병래

2004년 목포대학교 대학원 컴퓨터공학과 졸업(공학박사)
2005년 호남대학교 컴퓨터공학과 전임강사
2009년 ~ 현재 광주과학기술원 정보통신공학부 연구조교수
2012년 ~ 현재 제노테크(주) 대표
<주관심분야 : 정보보안, IDS, Neural Network, Cloud Computing, VoIP, NFC 등>

최명수

2009년 목포대학교 전자공학과 공학박사
2009년 목포대학교 해양텔레매틱스기술개발센터 박사후연구원
2010년 목포대학교 정보산업연구소 연구전임교수
2015년 ~ 현재 제노테크(주) 기업부설 연구소 연구소장
<주관심분야 : 정보보안, IDS, Neural Network, Cloud Computing, VoIP, NFC>

박 선

2007년 인하대학교 컴퓨터정보공학과 공학박사
2008년 호남대학교 컴퓨터공학과 전임강사
2010년 전북대학교 인력양성사업단 박사후 과정
2010년 목포대학교 정보산업연구소 연구전임교수
2013년 ~현재 광주과학기술원 연구조교수
<주관심분야 : 정보검색, 데이터마이닝, 해양IT정보융합, 클라우드 컴퓨팅, IoT, 스토리지 시스템>

김종원

1997년 University of Southern California 연구 조교수
1999년 Technology Consultant for VProtect Systems Inc.
2000년 Technology Consultant for Southern California Division of InterVideo Inc.
2001년 광주과학기술원 정보기전공학부 부교수
2008년 ~ 현재 광주과학기술원 정보통신공학부 교수
<주관심분야 : Future Internet, SDN & NFV, SDI>