AWS 클라우드에서의 웹 애플리케이션 호스팅

모범 사례

2010년 5월

Matt Tavis

요약

가용성과 확장성이 뛰어난 웹 호스팅은 복잡하고 비용이 많이들 수 있습니다. 기존의 확장 가능한 웹 아키텍처에서는 높은 수준의 신뢰성을 확보하기 위해 복잡한 솔루션을 구축하고, 고객 서비스의 질을 보장하기 위해 트래픽도 정확하게 예측해야 합니다. 트래픽 최고치가 일정 기간에만 밀집해 있고 트래픽 패턴의 변동 폭이 크면 하드웨어의 활용도가 낮아지게 되는데, 이렇게 고가의 운영 비용이 드는 하드웨어를 놀리게 되면 활용도가 떨어지는 하드웨어에 집행한 자본 투자의 효율도 낮아지게 됩니다. 이에 반해, Amazon Web Services는 트래픽 사용량이 급변하는 웹 애플리케이션에 필요한 신뢰성, 확장성, 보안, 고성능이라는 장점을 갖춘 인프라는 물론 실시간 고객 트래픽 패턴에 맞게 IT 비용을 최적화할 수 있는 탄력적 확장-축소형 인프라 모델도 제공합니다.

대상

본 백서는 인프라의 확장성을 확보하고 온-디맨드 컴퓨팅 수요를 해결하는 데 필요한 클라우드 컴퓨팅 서비스를 찾고 있는 IT 관리자 및 시스템 아키텍트들을 대상으로 합니다.

기존 웹 호스팅의 개요

웹 호스팅의 확장성은 이미 잘 알려진 주제이기 때문에, 기존 웹 호스팅 모델에 익숙한 사람들에게는 이 장에서 다루는 내용이 그리 놀랍지 않을 것입니다. 이 섹션에서는 클라우드에 웹 호스팅 아키텍처를 구현하는 일에 대해 논할 때 주로 비교 대상으로 등장하는 표준 웹 호스팅 아키텍처의 예제를 소개합니다.

기존 웹 애플리케이션 호스팅 아키텍처의 사례

아래는 기존 웹 호스팅 모델을 사용한 확장 가능한 웹 호스팅 아키텍처의 전형적인 예입니다.

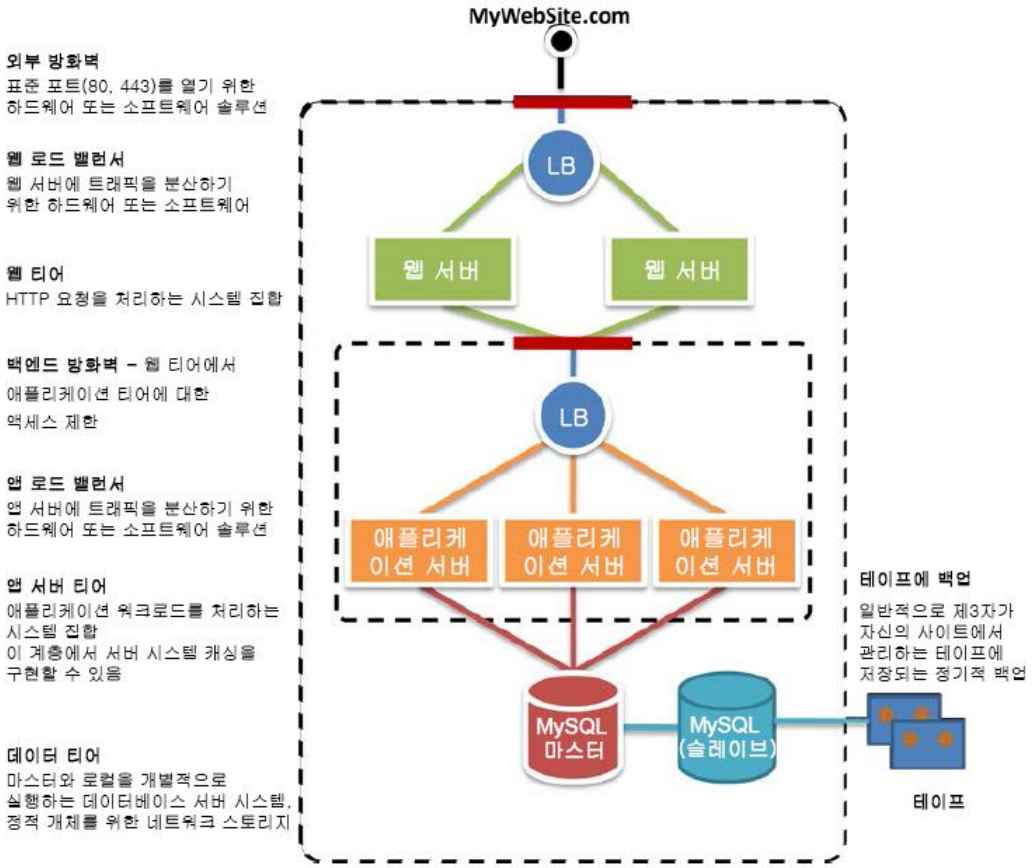

그림 1 - 기존 웹 애플리케이션 아키텍처

기존 웹 호스팅 아키텍처는 구조를 프레즌테이션, 애플리케이션, 퍼시스턴스 layer로 구분하는 일반적인 3 계층 웹 애플리케이션 모델을 중심으로 설계됩니다. 이 아키텍처는 프리젠테이션, 퍼시스턴스 또는 애플리케이션 layer에 호스트를 더하는 방식으로 확장할 수 있도록 설계되었으며 필요 성능, 장애 조치, 가용성 을 갖추고 있습니다. 다음 섹션에서는 이러한 아키텍처를 Amazon Web Services 클라우드에 이전할 때의 혜택과 옮기는 방법에 대해 살펴보겠습니다.

Amazon Web Services를 이용한 클라우드에서의 웹 애플리케이션 호스팅

기존 웹 호스팅 아키텍처(그림 1)는 몇 가지 적은 수정만으로 AWS 제품에서 제공하는 클라우드 서비스에 이식될 수 있습니다. 그보다 먼저 생각해보아야 할 것은 과연 이 어플리케이션을 클라우드 환경에 이전하는 것이 적절한지에 대한 고려입니다.

기존 애플리케이션 호스팅 솔루션을 AWS 클라우드로 이전하는 것에는 어떤 가치가 있을까요?

AWS가 일반적인 웹 애플리케이션 호스팅 문제를 해결하는 방법

여러분이 웹 애플리케이션의 운영을 담당한다면 수많은 인프라스트럭처 및 아키텍처 관련 문제들을 겪게 될 것입니다. AWS는 이에 대해 쉽고 원활한 솔루션들을 효율적인 비용으로 제공합니다. 다음은 기존 호스팅 모델 대신 AWS를 사용하는 것의 이점 중 일부입니다.

과도한 서버 대신 효율적 비용으로 트래픽 최고치를 처리

기존 호스팅 모델에서는 트래픽 급증에 대응하기 위해 서버를 미리 프로비저닝해 둬야 했으며 트래픽이 최고치까지 치솟지 않는 동안은 그 자원이 낭비됩니다. 반면 AWS로 웹 애플리케이션을 호스팅하면 서버를 추가하는 과정에서 온-디맨드 프로비저닝이 가능하므로 트래픽에 따라 사용 용량과 비용이 변화합니다.

예를 들어 다음의 그래프는 한 웹 애플리케이션이 하루중 오전 9시부터 오후 3시 사이에 트래픽 급증하며 그 외의 시간은 낮은 사용률을 보여줍니다. 실제 트래픽 추이에 기반한 오토 스케일링과 필요시에만 자원을 사용하는 프로비저닝 방식을 택하면 사용량과 비용을 50%이상 절감할 수 있습니다.

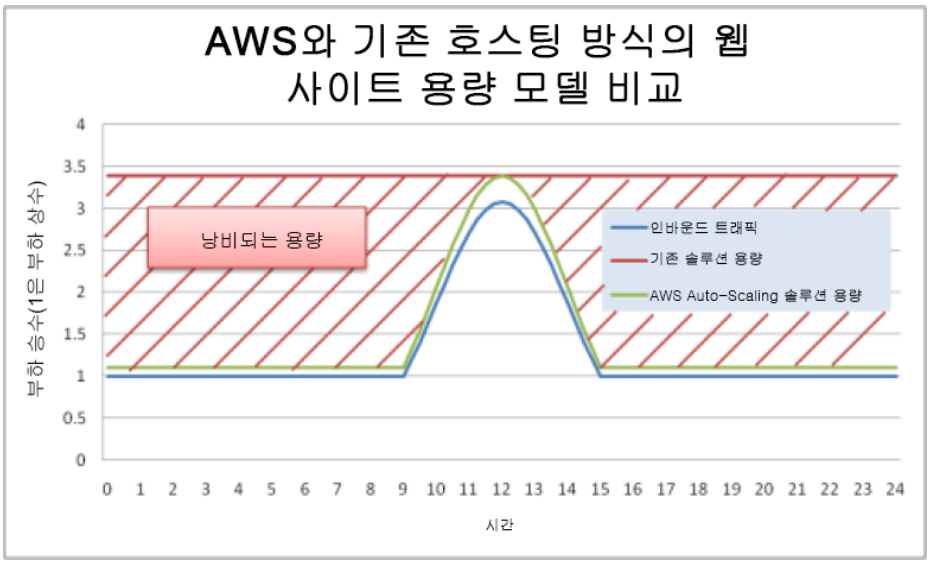

그림 2 - 기존 호스팅 모델에서 낭비된 용량의 사례

예측하지 못한 트래픽 피크를 처리하는 확장성 솔루션

이보다 더 끔찍한 일은 기존 호스팅 모델의 느린 프로비저닝 속도 때문에 예상하지 못한 트래픽 요청 급증에 제 때 대응하지 못하는 것입니다. 어떤 서비스가 예상치 못하게 유명 매체에서 언급된 후 트래픽 급증이 일어나 웹 애플리케이션에 장애가 발생했다는 식의 이야기를 많이 들어보셨을 겁니다. 예상치 못한 트래픽 급증이 발생한다고 해도 평소 일반적인 트래픽 증감에 맞춰 웹 애플리케이션을 확장 및 축소하도록 해 주는 온-디맨드 사용 구조는 그대로 단 몇 분 만에 새로운 호스트를 추가해 충분히 대응할 수 있습니다.

테스팅, 로드, 베타 및 사전 생산 환경을 위한 온-디맨드 솔루션

웹 애플리케이션 서비스를 위한 시스템 자원을 전통적 호스팅 환경에서 구성할 경우, 서비스를 위한 시스템 구성에서만 비용이 발생하는 것은 아닙니다. 웹 애플리케이션의 품질을 보장하기 위해서는 실제 서비스를 위한 컴퓨팅 환경을 구성하기 전에 개발 주기 단계마다 웹 애플리케이션을 최적화하기 위한 시험 서비스, 베타 및 테스트 환경을 구성할 필요가 있습니다. 이 테스팅 하드웨어들의 활용도를 극대화하기 위해 여러 가지 최적화 기술들을 적용해 본다고 하더라도 이 자원들이 정말로 적절하게 활용되는 경우는 매우 드뭅니다. 따라서 많은 고가의 하드웨어가 오랜 시간 동안 사용되지 않은 채로 놓여 있게 됩니다. AWS 클라우드를 사용할 경우, 테스트 시스템 자원 등을 필요에 따라 프로비저닝할 수 있으므로 정말 필요할 때만 이런 자원들을 쓰는 경제적 활용이 가능합니다. 또한 최종 버전의 부하 테스트를 수행하기 위해 사용자 트래픽을 시뮬레이션할 때도AWS 클라우드를 활용할 수 있으며 클라우드를 새로운 제품 출시를 위한 준비 환경으로 사용해 서비스 중단을 최소화한 채 애플리케이션 버전을 빠르게 업데이트하는 것도 가능합니다.

AWS 클라우드 아키텍처를 통한 웹 호스팅

아래는 기존 웹 애플리케이션 아키텍처에서 AWS 클라우드 컴퓨팅 인프라를 활용하는 방법을 나타낸 것입니다.

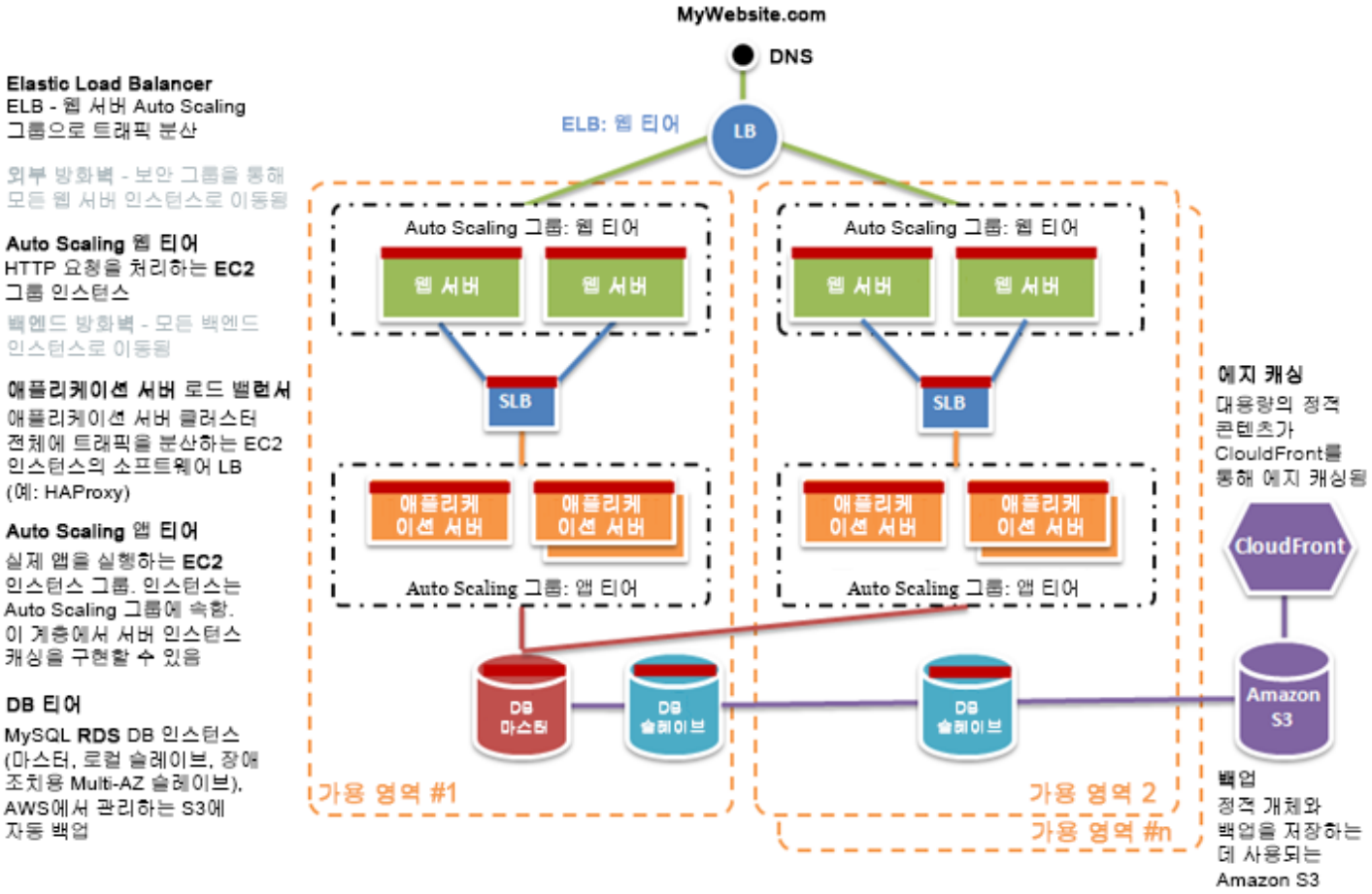

그림 3 - AWS에서 웹 호스팅 아키텍처의 사례

AWS 웹 호스팅 아키텍처의 핵심 구성 요소

이 섹션에서는 AWS 웹 호스팅 아키텍처의 핵심적 구성 요소 몇 가지와 기존 웹 호스팅 아키텍처와의 차이점을 설명합니다.

콘텐츠 전송

Amazon Web Service 클라우드 컴퓨팅 인프라에서도 엣지 캐싱은 중요한 요소입니다. 사용자의 웹 애플리케이션 인프라에 존재하는 모든 솔루션들이 AWS 클라우드에서 잘 작동하지만, AWS를 사용하면 고려할 수 있는 선택지의 수가 하나 늘어납니다. Amazon CloudFront 서비스[1]를 활용해 Amazon Simple Storage Service[2](Amazon S3)에 저장된 애플리케이션을 위한 콘텐츠를 엣지 캐싱하는 것입니다. Amazon EC2와 엣지 캐싱 솔루션을 함께 사용할 경우의 주된 이점은 로컬 엣지 캐시 Point-of-Presence(POP)를 통해 고객 관점에서의 애플리케이션 성능을 가속화할 수 있다는 것입니다. 고객에게 가장 가까운 지역에 해당 콘텐츠를 캐싱함으로써 스트리밍 또는 다운로드된 정적 콘텐츠(예: 플래시 동영상 또는 이미지)를 훨씬 빠르게 로딩합니다. 또 다른 이점은 CloudFront 또한 다른 AWS 서비스들과 마찬가지로 특별한 *조건이나 최소 의무 사용량, 계약 사항 등이 없는 사용구조를 갖고 있으므로* 필요에 따라 원하는 만큼의 양과 기간 동안 사용할 수 있다는 것입니다.

CNAME 및 Elastic IP를 이용한 공용 DNS의 관리

웹 애플리케이션을 AWS 클라우드로 이전하는 데에는 일부 DNS에 대한 변화가 필수적입니다. AWS는 공용 DNS 관리 서비스를 제공하지 않으므로 공용 인터넷 트래픽을 AWS 클라우드의 애플리케이션에 리다이렉트하기 위해서는 공용 DNS가 Elastic Load Balancing CNAM 또는 엘라스틱 IP 주소를 가리키도록 변경해야 합니다. 하지만 DNS는 하위 도메인에 대한 CNAME의 사용을 제한하기 때문에 루트 도메인(예: example.com)이 Elastic Load Balancer CNAME을 가리킬 수 없습니다. 시간이 지남에 따라 Elastic Load Balancer 뒤의 IP 주소가 변경될 수 있으므로 루트 DNS A-record를 Elastic Load Balancer CNAME 뒤의 IP에서 가리키도록 하는 것은 불가능하다는 것을 고려해야 합니다. 이에 대한 간단한 차선책은 동적으로 할당할 수 있는 정적 IP 주소인 Elastic IP를 애플리케이션에서 두개 이상의 EC2웹 서버에 지정하고, 이 웹 서버들이 웹 트래픽을 올바른 하위 도메인으로 리디렉트하게 함으로써 트래픽을 Elastic Load Balancer CNAME(예: www.example.com)으로 라우팅하도록 하는 것입니다. 공개 도메인 이름을 구매할 때 이용한 도메인 이름 공급자는 올바른 하위 도메인(예: www.example.com)에 대한 Elastic Load Balancer CNAME 신청을 위해, 그리고 루트 도메인 A-records에 대한 엘라스틱 IP 주소 목록을 설정하기 위해 간단한 체계를 제공해야 합니다.

[1] Amazon CloudFront – http://aws.amazon.com/cloudfront/
[2] Amazon S3 – http://aws.amazon.com/s3

호스팅 보안

기존 웹 호스팅 모델과는 달리 인바운드 네트워크 트래픽 필터링은 엣지에서 제한되기보다는 호스팅 수준에서 적용되어야 합니다. Amazon EC2는 인바운드 네트워크 방화벽과 유사한 *보안 그룹*이라 불리는 기능을 제공하여 EC2 인스턴스에 접속할 수 있는 프로토콜, 포트 및 소스 IP의 범위를 지정할 수 있도록 해 줍니다. 각각의 EC2 인스턴스는 하나 이상의 보안 그룹에 할당될 수 있으며, 적합한 트래픽을 각 인스턴스로 라우팅합니다. 보안 그룹은 EC2 인스턴스에 대한 액세스를 가진 특정 서브넷 또는 IP 주소만으로 구성될 수 있습니다. 또는 특정 그룹에 속한 EC2 인스턴스에 대한 액세스를 제한하기 위해 다른 보안 그룹을 참고할 수 있습니다. 예를 들어 AWS 웹 호스팅 아키텍처 사례(위의 그림)에서 클러스터 웹 서버의 보안 그룹은 모든 호스트에 대해 오직 TCP 80번, 443번(HTTP, HTTPS) 포트를 통한 접근만 허용하고, 직접 호스트를 관리하기 위해 어플리케이션 서버 보안 그룹에 속한 인스턴스들에 한해 22번(SSH) 포트로의 접근을 허용하도록 할 수 있습니다. 한편 애플리케이션 서버 클러스터의 보안 그룹은 웹 서버 보안 그룹에 접근을 허용해 웹 요청을 처리하도록 하고 사내 서브넷에 한정된 22번(SSH) 포트의 접속을 허용해 직접 호스트를 관리하도록 할 수 있습니다. 이 모델에서는 지원 엔지니어들이 기업 네트워크에서 애플리케이션 서버에 직접 로깅해 애플리케이션 서버 박스에서 다른 클러스터에 액세스할 수 있습니다. AWS의 보안에 대해 더 자세히 알고 싶으실 경우 AWS 보안 센터[3]에 방문해 주시기 바랍니다. AWS 보안 센터에서는 AWS의 보안 성능을 설명하는 보안 게시판, 인증 정보 및 보안 백서를 제공합니다.

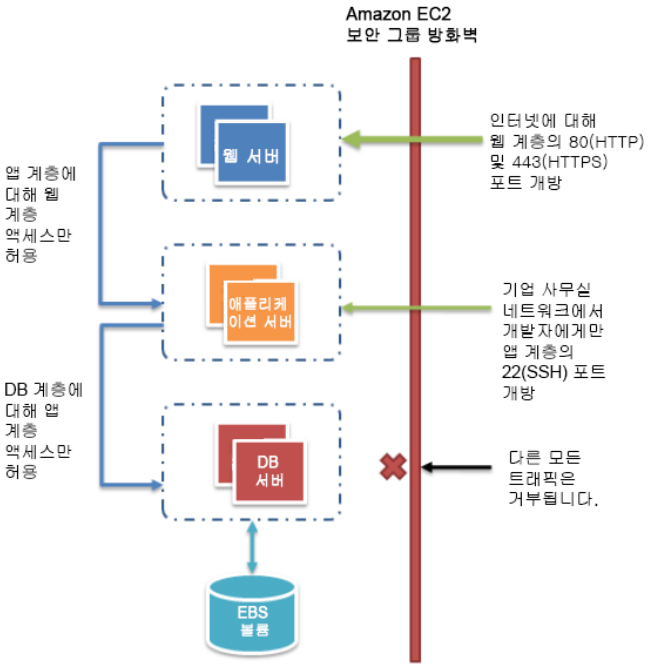

그림 4: 웹 애플리케이션에서의 보안 그룹

[3] AWS 보안 및 준수 센터 – http://aws.amazon.com/ko/security

클러스터 사이의 로드 밸런싱

하드웨어 로드 밸런서는 기존 웹 애플리케이션 아키텍처에서 사용되는 일반적인 네트워크 어플라이언스입니다. AWS는 호스팅 상태 점검, 다중 가용 영역에 걸친 EC2 인스턴스에 대한 트래픽 배포 및 로드 밸런싱 로테이션의 EC2 호스팅 추가 및 제거를 지원하는로드 밸런싱 솔루션인 Elastic Load Balancing[4] 서비스를 통해 이와 동일한 성능을 제공합니다. 사용자 임의의 설정이 가능한 Elastic Load Balancing은 영구 CNAME을 통해 예측 가능한 진입점을 제공하는 한편, 증감하는 트래픽 수요에 대응하기 위해 로드 밸런싱 용량을 동적으로 확장 및 축소할 수도 있습니다. Elastic Load Balancing 서비스는 어드밴스 라우팅 요구를 처리할 수 있도록 sticky 세션도 지원합니다. 애플리케이션이 어드밴스 로드 밸런싱 성능을 필요로 한다면 그 대안은 EC2 인스턴스(예: Zeus, HAProxy, nginx)에 소프트웨어 로드 밸런싱 패키지를 설치하여 사용하며, 로드 밸런싱이 적용된 이 EC2 인스턴스들에 Elastic IP를 지정해 DNS 변화를 최소화하는 것 입니다.

다른 호스팅 및 서비스의 검색

기존 웹 호스팅 아키텍처와의 또 다른 차이점은 대부분의 호스트가 동적 IP 주소를 갖게될 것이라는 점입니다. 모든 EC2 인스턴스는 공개 및 사설 DNS 항목을 둘 다 가질 수 있으며 인터넷 상에서 주소를 부여할 수 있지만, DNS 항목 및 IP 주소는 인스턴스의 시작에 따라 동적으로 할당되며 수동으로 할당될 수 없습니다. 정적 IP (AWS 용어로는 Elastic IP)는 시작과 동시에 실행중인 인스턴스에 할당될 수 있지만, Elastic IP를 갖춘 AWS 클라우드 상의 호스트들만이 네트워크 커뮤니케이션에 대한 일관된 endpoint를 갖습니다. Elastic IP는 마스터 데이터베이스, 중앙 파일 서버 및 EC2 호스팅 로드 밸런서 등 일관된 endpoint를 필요로 하는 인스턴스 및 서비스용으로 사용되어야 합니다. 웹 서버와 같이 쉽게 확장-축소할 수 있는 인스턴스 유형은 위에 언급된 영구적인 서비스를 통해 자체 동적 endpoint에서 발견할 수 있도록 만들어져야 합니다. 이에 대한 간단한 솔루션은 인스턴스의 구성 및 요구되는 네트워크 서비스 endpoint를 중앙집중식으로 유지하는 것입니다. 그리고 나서 부트스트랩핑 스크립트를 통해 인스턴스의 시작 동안 사용할 수 있는 일관된 Elastic IP 기반 endpoint로 필요 시에 액세스할 수 있습니다. 대부분의 웹 애플리케이션 아키텍처에는 항상 운영되는 데이터베이스 서버가 있는데, 일반적으로 이 서버가 이런 유형의 구성 정보의 저장소가 됩니다. 염두에 둬야 할 것 중 하나는 EC2로 인프라를 구성해 지속적 서비스를 운영할 경우 비용 절감을 위해서 예약 인스턴스[5]를 고려해 볼 필요가 있다는 것입니다. 예약 인스턴스를 사용하면 새롭게 추가된 호스트가 부트스트랩핑 단계의 통신을 위해 필요한 endpoint 목록을 데이터베이스로부터 요청할 수 있습니다. 인스턴스가 시작될 때 이 데이터베이스의 위치를 유저 데이터[6] 형태로 각 인스턴스에 전달할 수 있습니다. 이 외에도 잘 정의된 endpoint에서 사용 가능한 고가용 서비스인 SimpleDB 서비스를 사용해 구성 정보를 저장하고 및 유지할 수 있습니다.

---

[4] Amazon Elastic Load Balancing – http://aws.amazon.com/elasticloadbalancing

[5] 예약 인스턴스 – http://aws.amazon.com/ec2/reserved-instances/

[6] 사용자 데이터 – http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/index.html?ApiReference-ItemType-RunInstancesType.html

웹 애플리케이션 내에서의 캐싱

기존 웹 애플리케이션 아키텍처 내에서 사용되는 소프트웨어 기반 캐싱 솔루션은 AWS 클라우드에서도 거의 비슷하게 사용됩니다. 캐싱 소프트웨어 솔루션을 사용해 EC2 인스턴스를 구축하기만 해도 충분히 AWS 클라우드에서 캐싱을 구현할 수 있습니다. 웹 및 애플리케이션 layer 캐싱을 이러한 방식으로 처리할 수 있으며 데이터베이스에서의 중앙집중식 구성으로 웹 및 애플리케이션 서버가 적합한 캐시 서버를 찾도록 할 수 있습니다.

데이터베이스 구성, 백업 및 failover

많은 웹 애플리케이션들이 특정 형태의 지속성을 가지며, 대개 데이터베이스 형식입니다. AWS는 클라우드에서 데이터베이스를 사용할 수 있도록 해 주는 주요 솔루션 두 가지를 제공하는데, Amazon EC2 인스턴스에서 관계형 데이터베이스(RDBMS)를 직접 호스팅하거나, 호스팅된 RDBMS 솔루션인 Amazon Relational Database Service(RDS)를 사용하는 것입니다. AWS는 EC 2에서 MySQL, Oracle, SQLServer, DB2 등 다양한 데이터베이스 솔루션을 쓸 수 있도록 지원하고 있습니다. RDS의 사용은 개발자에게 친숙한 연결 방법(예: ODBC 또는 JDBC)을 제공하며, 또한 웹 서비스 API를 통한 간편한 관리 기능도 제공합니다. 간단한 API 호출을 통해 스토리지 추가나 데이터 백업, 더 큰 EC2 인스턴스로의 데이터 이전 등 일반적인 작업들을 자동화할 수 있습니다. 기존의 호스팅 모델과 마찬가지로 클라우드의 데이터베이스 솔루션 또한 백업 및 failover를 지원할 수있도록 마스터 인스턴스와 슬레이브 인스턴스 모두를 갖춰야 합니다. EC2에서 데이터베이스를 호스팅하는 AWS 고객들은 EC2 인스턴스 상에서 읽기 전용 복제본의 미러링 및 로그 전달을 통한 항상 준비된 패시브 슬레이브 등 다양한 마스터/슬레이브 및 복제 모델을 사용하고 있습니다. 웹 애플리케이션은 대개 특정 데이터베이스 백업 및 실패 모델을 채택하고 있는데, 대부분의 경우 AWS 클라우드에서 이러한 모델을 쉽게 복제해 구현할 수 있습니다. RDS를 사용하는 AWS 고객에게는 간단한 API 호출을 통한 기본 백업 및 failover 매커니즘을 제공합니다. 데이터베이스의 가용성을 극대화하기 위해 failover를 위한 멀티 슬레이브를 여러 곳의 가용 영역에 운영하는 방식의 RDBMS 구성이 권장됩니다. 각 가용 영역들은 리전의 가용성을 최대한 보장하기 위해 동일 리전 내 다른 가용 영역과 완전히 분리되어 있으므로, 가용 영역을 두 개 이상 사용하는 것은 백업 데이터센터를 보유하는 것과 매우 유사합니다. RDS를 사용하는 AWS 고객은 자동으로 다른 가용 영역에 바로 사용될 수 있는 예비 슬레이브 인스턴스를 배포하는 다중 가용 영역(Multi-AZ) 기능의 장점을 누릴 수 있습니다.

RDS를 사용하지 않고 EC2에서 직접 데이터베이스를 실행할 때 더 고려해 봐야 할 사항은 내결함 지속 스토리지의 사용이 가능한지 생각해 보는 것입니다. 이 조건을 만족시킬 수 있도록 Amazon EC2에서 데이터베이스를 실행할 때에는 Amazon Elastic Block Storage(Amazon EBS) 볼륨을 활용하도록 권장됩니다. EBS는 데이터베이스가 실행 중인 EC2 인스턴스에 대한 NAS(network attached storage)와 유사한 역할을 합니다. EC2 인스턴스에서 데이터베이스를 실행할 때, 모든 데이터베이스의 데이터 및 로그는 데이터베이스 호스트가 실패하더라도 데이터와 로그를 사용 가능하도록 Amazon EBS 볼륨에 저장해야 합니다. 이렇게 하면 호스트가 실패하는 경우에도 새로운 EC2 인스턴스를 생성하고 여기에 사용하던 Amazon EBS 볼륨을 붙여 데이터베이스가 실패 전의 위치에서 데이터를 그대로 액세스 할 수 있습니다. 또한 Amazon EBS 볼륨은 가용 영역 내에서 자동으로 이중화를 제공해 가용성을 단순히 디스크를 사용하는 것 이상으로 높입니다. 단일 Amazon EBS 볼륨의 성능이 필요한 데이터베이스 요건을 만족시키지 못한다면 볼륨들을 데이터베이스에 맞게 IOPS 성능을 향상시키도록 스트라이프할 수 있습니다. RDS를 사용할 경우 RDS 서비스가 직접 Amazon EBS 볼륨을 관리합니다. RDS를 사용할 경우 Amazon EBS 볼륨의 관리는 RDS 서비스로 합니다.

AWS는 EC2 상에서의 관계형 데이터베이스 사용 지원 외에도, 고정된 스키마 없이도 쿼리 질의와 데이터 인덱싱이 가능한 작은 사이즈의 고가용성 및 내결함성 핵심 비관계형 데이터베이스 서비스를 제공하는 SimpleDB 서비스도 지원합니다. 인덱싱이 많이 되어 있으며 유연한 대형 단일 스키마 테이블의 데이터 액세스가 필요한 상황에서 SimpleDB 데이터베이스의 좋은 대안이 될 수 있습니다. SimpleDB 사용의 예로는 구성 관리 데이터, 제품 카탈로그 및 세션 데이터 등이 있습니다. 이 외에도 EC2는 Cassandra, CouchDB, MemcacheDB 등 NoSQL 진영의 떠오르는 신규 기술들을 사용한 호스팅이 가능합니다.

데이터 및 자산에 대한 스토리지와 백업

AWS 클라우드에는 웹 애플리케이션 데이터 및 자산을 저장, 액세스 및 백업하는 다양한 방법들이 있습니다. Amazon Simple Storage Service(Amazon S3)는 고가용성 이중화 객체 저장 공간이며, 이미지나 동영상, 혹은 이 외의 정적 컨텐츠 등 정적이거나 느리게 변화하는 객체를 저장할 수 있는 탁월한 스토리지 솔루션입니다. Amazon S3를 CloudFront 서비스와 연동해 사용하면 이러한 데이터를 엣지 캐싱하거나 스트리밍하는 것도 가능합니다.
스토리지와 직접 붙여 쓸 수 있는 파일 시스템의 경우 EC2 인스턴스에 Amazon Elastic Block Storage 볼륨을 붙여 사용할 수 있으며, 이는 EC2 인스턴스 가동을 위한 장착형 디스크와 같은 역할을 수행합니다. Amazon EBS는 데이터베이스 파티션이나 애플리케이션 로그 등 블록 스토리지로 액세스할 필요가 있거나 실행 중인 인스턴스와 별개의 지속성을 요구하는 데이터에 적합합니다. EC2 인스턴스와 별개로 유지되는 지속성에 더해 Amazon EBS 볼륨의 스냅샷을 생성 Amazon S3에 저장하는 것도 가능하므로, 실행 중인 인스턴스 데이터를 백업하는 데 사용할 수 있습니다. EBS 스냅샷은 데이터 증분 백업이 가능하며, 따라서 자주 스냅샷을 생성할수록 스냅샷에 걸리는 시간은 줄어듭니다. Amazon EBS 볼륨은 1TB까지 생성할 수 있으며, 다중 Amazon EBS 볼륨을 스트라이핑해서 더 큰 볼륨을 만들거나 I/O 성능을 향상시키는 것도 가능합니다. Amazon EBS 스냅샷의 또다른 유용한 특징은 여러 개의 Amazon EBS 볼륨으로 복제하거나 복제된 볼륨을 실행 중인 다른 인스턴스에 첨부하여 사용이 가능하다는 점입니다.

플릿 Auto Scaling

AWS 웹 아키텍처와 기존 호스팅 모델 간의 주요 차이점 중 한 가지는 증감하는 트래픽에 대응할 수 있도록 웹 애플리케이션 플릿을 수요에 맞게 동적으로 확장할 수 있는 기능입니다. 일반적으로 기존 호스팅 모델에서는 예상되는 트래픽의 양에 맞춰 호스트의 양을 미리 프로비저닝하는 트래픽 예측 모델을 사용합니다. AWS 클라우드 아키텍처에서는 플릿을 확장 및 축소하는 트리거 세트에 기초하여 인스턴스를 빠르게 프로비저닝할 수 있습니다. Amazon Auto Scaling은 서버 용량 그룹을 생성해 실시간으로 수요에 맞춰 늘리거나 줄일 수 있도록 해 줍니다. Auto Scaling은 데이터 지표를 살피는 CloudWatch, 그리고 부하를 분산하기 위해 호스트를 추가하거나 제거해 주는 Elastic Load Balancing 서비스와 직접 연동됩니다. 예를 들어, 웹 서버들이 일정 시간 동안 80% 이상의 CPU 사용량을 보고할 경우 신속하게 웹 서버를 추가 구축할 수 있으며, 이는 자동으로 Elastic Load Balancer에 추가되어 즉시 부하 분산에 쓰입니다. AWS 웹 호스팅 아키텍쳐 모델에 나타나 있듯이, 아키텍처의 각 레이어마다 다른 오토스케일링 그룹을 만들어 레이어들이 각기 독립적으로 확장 및 축소되도록 할 수 있습니다. 예를 들어, 웹 서버 auto-scaling 그룹은 네트워크 I/O 상에서 확장-축소를 트리거 할 수 있으며, 애플리케이션 서버 auto-scaling 그룹은 CPU 사용량을 확장-축소할 수 있습니다. 최대값과 최소값을 지정해 상시 가용성을 보장하고 그룹 내 사용량을 제한하는 것도 가능합니다. Auto Scaling 트리거를 조정해 해당 레이어에 맞게 전체 플릿을 확장 및 축소하도록 함으로써 리소스 사용량을 실제 트래픽 수요에 맞추도록 할 수도 있습니다. Auto Scaling 서비스를 사용하는 방법 외에도 직접 EC2 API를 사용해 EC2 플릿을 손쉽게 확장할 수 있으므로 인스턴스를 시작 혹은 중단하거나 검사하는 것도 가능합니다.

AWS를 사용한 failover

기존 웹 호스팅 대신 AWS를 사용할 때의 또 다른 주요 이점은 가용 영역이 존재해 웹 애플리케이션 개발자들이 인스턴스를 배포하기 위해 다양한 위치에 쉽게 액세스할 수 있다는 것입니다. 가용 영역은 다른 가용 영역에 오류가 발생할 경우 해당 가용 영역의 오류로부터 분리되도록 설계된 별개의 위치로, 저렴하고 지연 시간도 낮은 네트워크를 통해 동일 리전의 다른 가용 영역에 연결되어 있습니다. AWS 웹 호스팅 아키텍처에서 볼 수 있듯 여러 가용 영역에 EC2 호스트를 분산해 웹 애플리케이션에 장애가 발생하지 않도록 하는 방법을 권장합니다. 실패가 발생할 경우를 대비해 단일 접속 지점을 여러 가용 영역에 나눠 이전할 수 있도록 프로비저닝하는 것이 중요합니다. 예를 들어, 예상치 못한 장애 상황에서도 데이터의 지속성과 접근성을 보장할 수 있도록 슬레이브 데이터베이스를 기존의 가용 영역 외에 다른 가용 영역에도 설치해 둘 것을 권장합니다.

기존의 웹 애플리케이션을 AWS 시스템으로 이전하는 과정에서 종종 구조적인 변화가 필요할 수도 있지만, AWS 클라우드를 통해 확장성과 안정성, 그리고 비용 효율을 제고할 수 있다는 것을 감안한다면 충분한 가치가 있습니다.

웹 호스팅에 대한 AWS 사용 시 핵심 고려 사항

AWS 클라우드 시스템에는 기존 호스팅 모델과는 다른 몇 가지 주요 차이점들이 있습니다. 이전의 섹션에서는 클라우드에서 웹 애플리케이션을 배포할 때 고려해야 할 다양한 중점들을 살펴봤습니다. 이번 섹션에서는 애플리케이션을 클라우드로 가져올 때 고려해야 할 설계상의 주요 변화를 살펴보겠습니다.

물리적 네트워크 어플라이언스의 제거

AWS 클라우드에서는 물리적 네트워크 어플라이언스를 사용할 수 없습니다. 예를 들어 여러분이 AWS에 올리는 애플리케이션에 쓸 방화벽이나 라우터, 로드 밸런서 등은 더 이상 물리적 디바이스의 형태를 띠지 않으며, 그보다는 소프트웨어 솔루션으로 대체될 필요가 있습니다. 로드 밸런싱(Zeus, HAProxy, nginx, Pound 등)이나 VPN 연결 설정(OpenVPN, OpenSwan, Vyatta 등) 등 분야에 따라 다양한 종류의 엔터프라이즈급 소프트웨어 솔루션들이 존재합니다. 이는 AWS 클라우드 상에서 실행할 수 있는 것을 제한하는 것이 아니라, 오늘날 이런 디바이스들을 사용한다면 애플리케이션 설계에 변화가 있어야 한다는 것을 의미합니다.

어디에나 있는 방화벽

기존 호스팅 모델에서는 간단한 DMZ를 구축해 호스트들 간의 오픈 커뮤니케이션 환경을 마련했지만, AWS는 모든 호스트를 차단할 수 있는 보다 안전한 모델을 지향합니다. AWS 상의 애플리케이션 배포를 계획하는 단계에서 고려해 봐야 할 것 중 하나는 호스트 간의 트래픽을 분석하는 것으로, 이로써 정확히 어떤 포트가 열려 있어야 하는지에 대한 결정을 내릴 수 있습니다. 아키텍처의 각 호스트별 유형에 따라 EC2 내 보안 그룹을 생성할 수 있으며, 간단하게 여러 계층의 보안 모델을 다양하게 생성해 아키텍처 내의 호스트 사이에 최소한의 액세스만 가능하도록 설정할 수도 있습니다.

여러 데이터 센터 사용하기

EC2를 사용할 때 가용 영역을 여러 개 쓴다면 이는 데이터 센터를 여러 곳 사용하는 것으로 생각해야 합니다. 이들은 논리적으로, 그리고 물리적으로 구분되어 있으며 고가용성 및 안정성을 확보하기 위해 위해 여러 데이터 센터에 애플리케이션을 배포할 수 있도록 해 주는 사용하기 쉬운 모델을 제공합니다.

한시적인 자동증감 호스트

애플리케이션을 AWS에 올릴 때 고려해 봐야 할 기존 웹 호스팅 모델과 AWS의 차이점들은 여러 가지가 있겠지만 그 중 가장 중요한 것은 EC2 호스트가 자동으로 증감될 가능성이 있기 때문에 그 수명이 한시적이라는 사실입니다. AWS 클라우드에서 작동하도록 애플리케이션을 설계할 땐 호스트가 항상 사용 가능할 것이라고 생각하지 말아야 하며, 실패가 발생할 시엔 Amazon EBS 볼륨 상에 있는 데이터 외의 모든 로컬 데이터가 유실된다는 것을 유념해야 합니다. 그리고 새로운 호스트를 추가할 때는 해당 호스트의 IP 주소나 가용 영역 내에서의 위치에 대해 임의로 추측해서는 안됩니다. 이 때문에 기존의 방식보다 유연한 구성 모델을 짜고 호스트를 부트스트래핑할 때도 견고한 접근 방식을 취할 필요가 있지만, 이러한 방법들이야말로 확장성이 높고 내결함성을 갖춘 애플리케이션을 설계하고 실행하는데 반드시 필요한 요소라고 할 수 있습니다.

결론

클라우드로의 웹 애플리케이션 이전을 검토할 때 고려해 봐야 할 사항들은 매우 많습니다. 하지만 효율적인 비용으로 여러분의 비즈니스와 함께 성장할 수 있는 높은 확장성과 내결함성을 갖춘 인프라를 가질 수 있다는 이점은 AWS 클라우드로 이전하는 과정에서의 노고를 보상하고도 남을 것입니다.

참고 문헌

- **시작 안내서 - 웹 애플리케이션 호스팅:** http://docs.amazonwebservices.com/gettingstarted/latest/wah/
- **시작 안내서 - 리눅스용 웹 애플리케이션 호스팅:** http://docs.amazonwebservices.com/gettingstarted/latest/wah-linux