리퀘스트 예측을 통한 효율적인 오토스케일링 스케쥴러 구조 연구

주경노, 윤찬현
한국과학기술원 전기및전자공학과
e-mail : {eu8198, chyoun}@kaist.ac.kr

A Study on Structure of Efficient Autoscaling Scheduler Using Request Prediction

Kyung-No Joo, Chan-Hyun Youn
Dept. of Electrical Engineering, KAIST

요 약

클라우드 기술이 발달하면서 사용자 요구를 만족하면서도 비용을 절감하기 위해 VM 의 개수를 자동으로 조절해주는 오토스케일링 기술이 부각되었다. 하지만 어떤 VM 을 추가할 것인지는 NP-Hard Problem 으로 휴리스틱하게 풀 수밖에 없다. 따라서 사용자의 실시간으로 변하는 요구에 바로 대처하지 못할 수 있다. 사용자 요구에 실시간적으로 대처하기 위해서는 사용자가 보내는 요청의 패턴을 읽고, 앞으로 올 요청을 미리 아는 기술이 필요하다. 이에 본 논문에서는 리퀘스트 예측을 통한 오토스케일링을 가능케 하도록 구조를 제안하고자 한다.

1. 서론

최근 클라우드 기술이 각광받으면서 여러 연구가 진행 중에 있다. 클라우드는 온디멘드 형태로 CPU 와 스토리지, 메모리 등을 제공할 수 있기 때문에 플랫폼 자체로 서비스가 가능한데, 클라우드 제공자에게 돈을 제공하면 클라우드의 컴퓨팅 자원을 받을 수 있다. 이는 현재 가지고 있는 컴퓨팅 자원으로 특정 업무를 처리하기에 부족함이 있을 때, 비싼 돈을 들여 추가적인 컴퓨팅 자원을 구매하거나 컴퓨팅 인프라스트럭쳐를 구축할 필요 없이 아웃소싱을 통해 효율적으로 컴퓨팅 자원을 보충할 수 있게 됨을 의미한다.

이 때, 클라우드 제공자는 사용자와 CPU, RAM, 응답 시간, I/O 등의 최소 요구치를 협상을 통해 결정해야 한다. 이를 서비스 레벨 협약서(Service Level Agreement: SLA)라고 부른다. 클라우드 제공자는 사용자에게 무조건 SLA 이상의 컴퓨팅 플랫폼을 제공해야 한다. 주어진 VM 들을 SLA 를 만족시키면서 최소의 가격으로 배치시킬 수 있다면 큰 비용 절감 효과를 볼 수 있을 것이다. 하지만 사용자의 요구는 실시간으로 변하기 때문에 이를 감지하고 알맞게 VM 을 새로 생성하거나 삭제하도록 해줄 기술이 필요하다. 이를 오토스케일링이라 부른다. 하지만 시간이 오래 걸리는 단점이 있다. 또한 변하는 사용자의 요구를 미리 예측하면 더욱 더 스마트한 스케일링이 가능할 것이다.

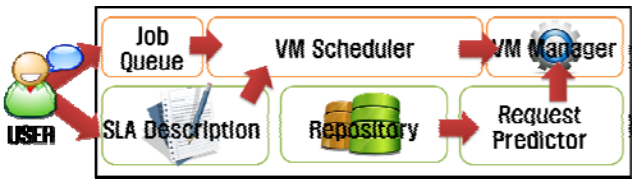
따라서 본 논문에서는 리퀘스트 예측을 통한 오토스케일링 스케줄러의 구조를 제안하고자 한다.

2. 오토스케일링 스케쥴러 가정

오토스케일러는 사용자 요구 레벨(SLA)에 맞춰 VM 들을 알맞게 배치시키기 위한 기술이며, 사용자의 요구를 미리 예측해 VM 을 미리 준비시켜둠으로서 시간을 더 절약할 수 있다. 이를 통해 코스트를 최소화시켜 시간과 비용을 절감시키는 효과가 있다. 이 때 본 논문에서 가정한 사항은 다음과 같다.

- 사용자가 한정적인 자본을 가지고 있어 VM 들을 마음껏 사용할 수 없다. 또한 SLA 를 만족하지 못할 수도 있다. 그 경우, 자본 내에서 최대의 컴퓨팅 파워를 계산해줘야 한다. [1][4]
- 가격은 시간마다 책정되며 시간이 차지 않은 경우 반올림해서 매겨진다. 따라서 가격이 넘어가기 전에 VM 을 종료하는 것은 좋은 정책이 될 수 있다. [1][3][4]
- 클라우드 인스턴스는 언제든지 생성할 수 있으나, 인스턴스를 사용하기까지 약간의 준비시간이 필요하다. 스케줄링을 할 때에는 이 준비시간도 고려해야 한다. [2][4]
- 서비스 업무는 여러 타입으로 나뉘며, 각각은 큐에 저장되어 FCFS 로 하나씩 하나씩 서비스된다. 이 때, 큐의 크기는 무한해 계속 요청을 받을 수 있다.[3]

3. 오토스케일링 스케쥴러 구조 설계



(그림 1) 오토스케일링 스케쥴러 모델

그림 1 은 본 논문에서 제안하는 스케줄러 구조도이다. 각 컴포넌트별 간단한 역할과 관계를 아래 정리하였다.

업무 큐(Job Queue)는 사용자가 아웃소싱한 워크로드들을 담아두는 큐이다. 사이즈가 무한하다고 가정되어 있으므로 버리는 워크로드 없이 모두 처리되게 된다.

VM 스케쥴러(VM Scheduler)은 SLA 와 VM 정보를 가지고 Job Queue 에서 넘어온 워크로드를 코스트를 최소로 하는 범위 내에서 VM 할당할 정책을 세운다. 자본 제한을 경계조건(boundary condition)으로 하고 가격을 최소화시키는 linear programming 방법을 사용해서 휴리스틱하게 정책을 구할 수 있다.

VM 관리자(VM Manager)는 VM 을 실제로 만들고, 수정하고, 삭제하는 등의 작업을 한다. 또한, 생성된 VM 들의 상태를 모니터링하고 리소스 정보를 시각화하는 역할도 한다. VM 스케쥴러에서 구한 정책을 실제로 시행하는 구간이다. 또한 이곳에서의 모니터링된 정보는 로그를 남겨 레파지토리에 기록된다.

SLA 기술 문서(SLA Description)는 사용자와 서비스 프로바이더 간에 정한 SLA 가 담겨있다. 단 VM 스케줄러가 계산할 수 있도록 CPU, RAM, i/o 등의 리소스 형태로 저장되어 있다. SLA 를 컴퓨팅 리소스로 변환시키는 것은 매우 어렵다. 이 논문에서는 경험적으로 데드라인 내에 워크로드를 끝마칠 수 있도록 컴퓨팅 리소스를 할당시켜주도록 했다.

레파지토리(Repository)에는 VM 사용 기록과 사용자가 요구한 업무들은 물론이고 사용자 개인 정보까지 저장되어 있는 장소이다. 여기 저장된 히스토리 요청을 토대로 리퀘스트 예측이 이루어지게 된다.

리퀘스트 프레딕터(Request Predictor)는 히스토리에 저장된 최근 한~두번의 요청을 토대로 다음에 올 요청을 예측하고, VM 관리자로 전달해 해당 VM 을 미리 켜 놓게 해 VM 인스턴스를 켜는 딜레이를 줄이도록 해 준다. 예상되는 요청은 앞의 두 요청을 토대로 다음 요청을 예상해주는 ARMA 필터를 이용해 간단히 구현하였으나 보간법이나 다른 필터를 사용해도 무방하다.

다시 말해, 사용자가 SLA 와 함께 워크로드를 주면, 워크로드가 저장된 큐에서 하나씩 꺼내져 스케쥴러에 들어간다. 스케쥴러에서는 휴리스틱한 방법을 통해 SLA 를 만족시키면서 최소의 비용으로 VM 을 할당시킬 정책을 짜며, 이를 VM 관리자에 전달시킨다. VM 관리자는 VM 을 생성하고 모니터링하는 역할을 하며 이 기록은 레파지토리에 저장된다. 한편, 레파지토리에 있는 데이터를 토대로 VM 을 키는 시간을 줄이기 위해 앞으로 나올 리퀘스트를 예측시켜주는 Predictor 를 거쳐 시간을 절약하게 된다.

4. 결론 및 제언

본 논문에서는 SLA 를 만족하면서 비용과 시간을 절약할 수 있도록 적절한 VM 을 아웃소싱시켜주는 오토스케일링 스케쥴러의 구조에 대해 소개하였다. 사용자가 보내는 요청의 패턴을 파악해 미리 VM 을 생성시켜 실시간으로 사용자 요청에 대처할 수 있도록 하였으며, 이는 상당한 시간적, 비용적 이득을 가져올 것으로 기대할 수 있다. 이렇게 만들어진 오토스케일링 스케쥴러는 이종 클라우드 환경에서 리소스를 협업하기 위한 클라우드 브로커에 그대로 사용될 수 있는 등 응용 분야도 많다. 논문에서 제안한 설계의 적합성을 실험하기 위한 테스트베드 구축과 검증, 실험하는 과정이 향후 이루어져야 할 것이다.

Acknowledgement

본 논문은 2012 년도 지식경제부의 재원으로 '개인 및 기업 맞춤형 서비스를 위한 개방형 모바일 클라우드용 통합개발환경 및 이기종 단말-서버 간 협업 기술 개발'과 BK21 사업의 지원을 받아 수행된 연구임.

참고문헌

[1] Ming Mao. Et al, Cloud Auto-scaling with Deadline and Budget Constraint, 2010

[2] Nilabja Roy. Et al, Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting, 2011

[3] Brian Dougherty. Et al, Model-driven auto-scaling of green cloud computing infrastructure, 2012

[4] Waheed Iqbal. Et al, SLA-Driven Adaptive Resource Management for Web Applications on a Heterogeneous Compute Cloud, 2009