고성능 컴퓨팅 렌즈

AWS Well-Architected 프레임워크

2019 년 12 월

Archived

This paper has been archived.

The latest version is now available at:

https://docs.aws.amazon.com/ko_kr/wellarchitected/latest/high-performance-computing-lens/welcome.html

고지 사항

고객에게는 본 문서에 포함된 정보를 독립적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공만을 위한 것이며, (b)사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c) AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 "있는 그대로" 제공됩니다. 고객에 대한 AWS 의 책임과 법적 책임은 AWS 계약서에 준하며 본 문서는 AWS 와 고객 간의 어떠한 계약도 구성하거나 수정하지 않습니다.

© 2019 Amazon Web Services, Inc. 또는 자회사. All rights reserved.

Archived

목차

Archived

요약

이 문서에서는 AWS Well-Architected 프레임워크의 **HPC(고성능 컴퓨팅) 렌즈**를 설명합니다. 일반적인 HPC 시나리오를 살펴보고, 모범 사례에 따라 워크로드 아키텍처를 설계하는 데 필요한 주요 요소를 알아봅니다.

Archived

소개

AWS Well-Architected 프레임워크는 AWS 에서 시스템을 구축할 때 내리는 의사 결정의 장점과 단점을 이해하는 데 도움이 됩니다.[1] 이 프레임워크를 사용하면 클라우드에서 안정적이고 안전하며 효율적이고 경제적인 시스템을 설계하고 운영하기 위한 설계 모범 사례를 알아볼 수 있습니다. 이 프레임워크는 모범 사례를 기준으로 아키텍처를 일관적으로 측정하고 개선 영역을 식별하는 방법을 제공합니다. 제대로 설계된 시스템을 갖추면 비즈니스 성공 가능성이 높아집니다.

이 "렌즈"에서는 AWS 클라우드 기반 **HPC(고성능 컴퓨팅)** 워크로드의 설계, 배포 및 아키텍처 구성 방법을 집중적으로 살펴봅니다. HPC 워크로드는 클라우드에서 특히 잘 실행됩니다. HPC 워크로드는 변동적이고 파열적인 성질이 있기 때문에 종량과금제 클라우드 인프라에 매우 적합합니다. 클라우드 인프라에서는 클라우드 리소스를 세부적으로 조정하고 클라우드 네이티브 아키텍처를 만들 수 있으므로 HPC 워크로드의 처리가 빨라지는 것이 당연합니다.

간결성을 위해 이 문서에서는 HPC 워크로드와 관련된 Well-Architected 프레임워크의 세부 정보만 다룹니다. 아키텍처를 설계할 때는 AWS Well-Architected 프레임워크 백서[2]의 모범 사례와 질문을 고려하는 것이 좋습니다.

이 문서는 CTO(최고 기술 책임자), 아키텍트, 개발자, 운영 팀 팀원 등 기술 업무 담당자를 위해 작성되었습니다. 이 백서의 내용을 확인하고 나면 클라우드 환경에서 HPC 를 설계하고 운영할 때 사용할 AWS 모범 사례와 전략을 파악할 수 있습니다.

Archived

정의

AWS Well-Architected 프레임워크는 운영 우수성, 보안, 안정성, 성능 효율성, 비용 최적화라는 5 가지 핵심 요소를 기반으로 합니다. 솔루션을 설계할 때는 비즈니스 상황에 따라 이러한 핵심 요소를 절충해야 합니다. 이러한 비즈니스 의사 결정에 따라 엔지니어링 우선 순위가 달라질 수 있습니다. 개발 환경에서는 안정성을 포기하는 대신 비용을 절감하도록 최적화하고, 미션 크리티컬 솔루션의 경우 비용 증가를 감수하면서 안정성을 기준으로 최적화할 수 있습니다. 보안 및 운영 우수성은 일반적으로 다른 기반과 절충 관계에 있지 않습니다.

이 백서에서는 커뮤니티에서 HTC(고처리량 컴퓨팅)라고 부르는 약결합 워크로드와 강결합 워크로드 간의 중요한 차이점에 대해 알아보고 서버 기반 설계와 서버리스 설계에 대해서도 살펴봅니다. 이러한 구분에 대한 자세한 설명은 시나리오 섹션을 참조하십시오.

AWS 클라우드의 일부 용어는 일반적인 HPC 용어와 다를 수 있습니다. 예를 들어 HPC 사용자는 서버를 "노드"로 지칭하지만 AWS 에서는 가상 서버를 "인스턴스"로 지칭합니다. HPC 사용자가 일반적으로 "작업"이라고 칭하는 것을 AWS 에서는 "워크로드"라고 합니다.

AWS 설명서에서 "vCPU"라는 용어는 "스레드" 또는 "하이퍼스레드"(또는 *반*물리적 코어)와 동의어로 사용됩니다. AWS 에서 HPC 애플리케이션의 성능 또는 비용을 평가할 때는 이 2 가지 요인을 고려해야 합니다.

클러스터 배치 그룹은 AWS 에서 네트워크 요구 사항이 가장 높은 애플리케이션의 컴퓨팅 인스턴스를 그룹화할 때 사용하는 방법입니다. 배치 그룹은 물리적 하드웨어 요소가 아닙니다. 배치 그룹은 모든 노드를 지연 시간이 짧은 네트워크 반경 내에 유지하는 단순한 논리 규칙입니다.

AWS 클라우드 인프라는 **리전** 및 **가용 영역**을 중심으로 구축됩니다. 리전은 다수의 가용 영역이 포함된 전 세계의 물리적 위치입니다. 가용 영역은 하나 이상의 개별 데이터 센터로 구성됩니다. 각 데이터 센터는 분리된 시설에 구축되며 이중화된 전력, 네트워킹 및 연결 기능을 갖추고 있습니다. HPC 워크로드의 특성에 따라 여러 가용 영역에 클러스터를 배치하여 안정성을 개선하거나 단일 가용 영역 안에 배치하여 지연 시간을 짧게 유지하는 것이 좋을 수 있습니다.

일반 설계 원칙

기존의 컴퓨팅 환경에서 아키텍처 관련 의사 결정은 종종 정적인 일회성 이벤트로 시행되며 경우에 따라서는 컴퓨팅 시스템의 생애 중에 주 소프트웨어 또는 하드웨어 업그레이드가

Archived

수행되지 않습니다. 프로젝트와 그 컨텍스트가 시시각각 달라지는 상황에서 이러한 초기 의사 결정은 시스템으로 변화하는 비즈니스 요구 사항을 충족하는 데 방해가 될 수 있습니다.

클라우드에서는 다릅니다. 프로젝트 확장에 따라 클라우드 인프라를 확장하여 기능을 지속적으로 최적화할 수 있습니다. 클라우드에는 온디맨드 방식의 자동화 및 테스트 기능이 있어 인프라 설계 변경에 따른 위험이 줄어듭니다. 지속적으로 시스템을 변경할 수 있으므로 프로젝트에서 표준 사례로 혁신 기술을 활용하는 것이 가능합니다.

Well-Architected 프레임워크는 클라우드에서 고성능 컴퓨팅을 제대로 설계할 수 있게 하는 다음과 같은 일반적인 설계 원칙을 제안합니다.

- **동적 아키텍처**: 평형 상태의 모델을 사용하는 고정된 정적 아키텍처와 비용 추정을 사용하지 마십시오. 아키텍처는 HPC 의 변화하는 수요에 맞춰 동적으로 확장되고 축소되어야 합니다. 아키텍처 설계 및 비용 분석은 HPC 활동의 자연적 주기와 명시적으로 일치해야 합니다. 예를 들어 시뮬레이션 집약적 작업 기간이 지난 후에는 작업이 설계 단계에서 랩 단계로 이동하므로 수요가 줄어들 수 있습니다. 또는 장기간의 꾸준한 데이터 누적 단계 후에는 대규모 분석 및 데이터 감소 단계가 이어질 수 있습니다. 많은 기존의 슈퍼컴퓨팅 센터와 달리 AWS 클라우드에서는 긴 대기열, 오래 실행되는 할당량 애플리케이션과 사용자 지정 및 소프트웨어 설치 제한을 방지할 수 있습니다. HPC 작업의 다수는 일정 간격으로 버스트가 발생하는 특성이 내재되어 있기 때문에 탄력성 및 종량과금제라는 클라우드 패러다임에 아주 적합합니다. AWS 의 탄력성 및 종량과금제 모델을 제공하는 AWS 에서는 시스템 초과 구독으로 인해 대기열에서 대기 중인 시스템 또는 유휴 시스템으로 인한 비용 낭비가 발생하지 않습니다. 컴퓨팅 클러스터 같은 환경을 지정된 시간의 지정된 요구 사항에 맞게 “적절한 크기”로 조정할 수 있습니다.
- **워크로드에 조달 모델 할당**: AWS 는 다양한 HPC 사용량 패턴을 고려하여 다양한 컴퓨팅 조달 모델을 제공합니다. 올바른 모델을 선택하면 필요한 용량에 대해서만 요금을 지불할 수 있습니다. 예를 들어 연구 기관에서는 동일한 기상 예측 애플리케이션을 여러 방법으로 실행할 수 있습니다.
    - 학술적 연구 프로젝트의 경우 다수의 파라미터 범위 및 집합을 사용하여 기후 변수의 역할을 조사합니다. 이러한 시뮬레이션에서 중요한 것은 비용이며 속도는 그리 중요하지 않습니다. 따라서 Amazon EC2 스팟 인스턴스가 아주 적합합니다. 스팟 인스턴스에서는 Amazon EC2 의 미사용 용량을 온디맨드에 비해 최대 90% 할인된 요금으로 사용할 수 있습니다.

Archived

  - 산불 시기에는 지역의 바람을 분 단위로 예측하여 소방관의 안전을 보장해야 합니다. 시뮬레이션이 지연될 때마다 안전하게 대피할 수 있는 가능성이 줄어듭니다. 이러한 시뮬레이션에서 분석 버스트를 수용하고 중단 없이 결과를 받으려면 온디맨드 인스턴스를 사용해야 합니다.
  - 매일 아침에 실행된 기상 예측은 오후에 TV 방송을 통해 제공됩니다. 이 경우 예약 인스턴스를 예약하면 필요한 용량을 매일 정시에 사용할 수 있습니다. 이 요금 모델은 온디맨드 인스턴스보다 할인된 요금을 제공합니다.

- **데이터에서 시작**: 아키텍처 설계를 시작하기 전에 데이터에 대해 확실히 아는 것이 중요합니다. 데이터의 오리진, 크기, 속도 및 업데이트를 고려하십시오. 성능 및 비용을 전체적으로 최적화할 때는 컴퓨팅을 중심으로 데이터 고려 사항을 감안해야 합니다. AWS 는 데이터에서 대부분의 가치를 추출할 수 있는 데이터 시각화를 비롯하여 강력한 데이터 및 관련 서비스 오퍼링을 제공합니다.
- **자동화를 통해 아키텍처 실험 간소화**: 코드를 통한 자동화를 활용하면 수작업 없이 저렴한 비용으로 시스템을 생성하고 복제할 수 있습니다. 코드의 변경 사항을 추적하고, 그 영향에 대한 감사를 시행하고, 필요한 경우 이전 버전으로 되돌릴 수 있습니다. 인프라를 사용한 실험이 쉽기 때문에 성능 및 비용에 맞춰 아키텍처를 최적화할 수 있습니다. AWS 가 제공하는 AWS ParallelCluster 같은 도구를 활용하면 HPC 클라우드 인프라를 코드로 처리하여 손쉽게 시작할 수 있습니다.
- **협업 지원**: HPC 작업은 협업 환경에서 수행되는 경우가 많으며 때로는 전 세계의 여러 국가에 걸쳐 수행되기도 합니다. 즉각적인 협업 후에는 더 넓은 범위의 HPC 및 과학 커뮤니티와 방법 및 결과를 공유하는 경우가 많습니다. 따라서 공유될 수 있는 도구, 코드 및 데이터와 공유 대상을 사전에 고려하는 것이 중요합니다. 전송 방법을 이 설계 프로세스에 포함해야 합니다. 예를 들어 AWS 에서 워크플로는 다양한 방법으로 공유될 수 있습니다. AMI(Amazon Machine Image), Amazon Elastic Block Store(Amazon EBS) 스냅샷, Amazon Simple Storage Service(Amazon S3) 버킷, AWS CloudFormation 템플릿, AWS ParallelCluster 구성 파일, AWS Marketplace 제품 및 스크립트를 사용할 수 있습니다. AWS 는 협업을 통해 HPC 문제를 해결하는 데 필요한 보안 및 협업 기능을 완벽하게 활용할 수 있는 탁월한 환경을 제공합니다. 이 환경은 선별된 그룹 내의 안전한 공유 또는 더 광범위한 커뮤니티와의 공개적인 공유를 지원하여 컴퓨팅 솔루션 및 데이터 세트의 영향력을 극대화합니다.
- **클라우드 네이티브 설계 사용**: 워크로드를 AWS 로 마이그레이션할 때 온프레미스 환경을 굳이 복제할 필요는 없습니다. 또한 이러한 복제는 최선의 선택이 아닙니다. AWS 는

Archived

새로운 설계 패턴 및 클라우드 네이티브 솔루션을 사용한 새로운 방식으로 HPC 워크로드를 실행할 수 있는 방대하고 심층적인 서비스를 제공합니다. 예를 들어 각 사용자 또는 그룹은 개별 클러스터를 사용할 수 있으며 이러한 개별 클러스터의 크기를 로드에 따라 독립적으로 조정할 수 있습니다. 사용자는 AWS Batch 같은 관리형 서비스 또는 AWS Lambda 같은 서버리스 컴퓨팅을 사용하여 기반 인프라를 관리할 수 있습니다. 기존의 클러스터 스케줄러를 사용하는 대신 워크로드에 필요한 경우에만 스케줄러를 사용하는 것이 좋습니다. 클라우드에서 HPC 클러스터는 영구적일 필요가 없으며 일시적인 리소스로 존재할 수 있습니다. 클러스터 배포를 자동화하면 한 클러스터를 종료하고 동일하거나 다른 파라미터로 새 클러스터를 빠르게 시작할 수 있습니다. 이 방법을 사용하면 필요한 만큼 환경을 만들 수 있습니다.

- **실제 워크로드 테스트**: 클라우드에서 프로덕션 워크로드의 성능을 확인할 수 있는 유일한 방법은 클라우드에서 테스트하는 것입니다. 대부분의 HPC 애플리케이션은 복잡합니다. 또한 애플리케이션의 메모리, CPU 및 네트워크 패턴을 단순한 테스트로 축소할 수 없습니다. 애플리케이션의 인프라 요구 사항은 모델에 사용되는 애플리케이션 해결기(수학적 메서드 또는 알고리즘), 모델의 크기 및 복잡성 등에 따라 달라집니다. 이러한 이유로 일반적인 벤치마크는 실제 HPC 프로덕션 성능을 예측하는 데 있어서 신뢰할 수 있는 기준이 되지 못합니다. 소규모의 벤치마크 세트 또는 "간단한 문제"를 사용하여 애플리케이션을 테스트하는 경우에도 가치가 거의 없기는 마찬가지입니다. AWS 에서는 실제로 사용한 것에 대한 요금만 지불하므로 자체 대표 모델을 사용하여 현실적인 개념 증명을 수행하는 것이 가능합니다. 클라우드 기반 플랫폼의 가장 큰 장점 중 하나는 마이그레이션 전에 실물 크기의 실제 테스트를 수행할 수 있다는 것입니다.
- **결과 도출 시간과 비용 절감 간의 균형**: 가장 큰 의미가 있는 파라미터인 시간과 비용을 사용하여 성능을 분석하십시오. 시간이 중요하지 않은 워크로드의 경우 비용 최적화에 초점을 맞춰야 합니다. 시간이 중요하지 않은 워크로드에 있어서 가장 비용이 저렴한 방법은 일반적으로 스팟 인스턴스입니다. 예를 들어 내년에 있을 컨퍼런스 전에 많은 수의 랩 측정값을 분석해야 하는 경우 스팟 인스턴스를 사용하면 고정된 연구 예산 내에서 최대한 많은 수의 측정값을 분석하는 데 도움이 될 수 있습니다. 반대로, 비상 대응 모델링 같은 시간이 중요한 워크로드의 경우 비용 최적화를 성능 최적화로 바꾸고 인스턴스 유형, 조달 모델 및 클러스터 크기를 선택하여 가장 짧고 즉각적인 실행 시간을 달성할 수 있습니다. 플랫폼을 비교할 때는 비 컴퓨팅 측면(예: 리소스 프로비저닝, 데이터 준비 또는 기존 환경의 경우 작업 대기열에 소요되는 시간)을 포함한 전체 솔루션 시간을 고려하는 것이 중요합니다.

Archived

시나리오

일반적으로 HPC 사례는 병렬 처리 기술이 요구되는 복잡한 계산 문제입니다. 제대로 설계된 HPC 인프라는 계산 중에 성능을 유지하여 이러한 계산을 지원할 수 있습니다. HPC 워크로드는 유전체학, 컴퓨터 화학, 금융 위험 모델링, 컴퓨터 응용 공학, 기상 예측 및 진원 영상 같은 기존 애플리케이션과 기계 학습, 딥 러닝 및 자율 주행 같은 신흥 애플리케이션 전반에 걸쳐 존재합니다. 그럼에도 불구하고 이러한 계산을 지원하는 기존의 그리드 또는 HPC 클러스터는 특정 워크로드에 최적화된 일부 클러스터 속성이 포함될 뿐, 그 아키텍처가 말도 안 될 정도로 유사합니다. AWS 에서는 네트워크, 스토리지 유형, 컴퓨팅(인스턴스) 유형 및 심지어 배포 방법을 전략적으로 선택하여 특정 워크로드의 성능, 비용 및 사용 편의성을 최적화할 수 있습니다.

HPC 는 동시에 실행되는 병렬 프로세스 간의 상호 작용 수준에 따라 약결합 워크로드와 강결합 워크로드라는 2 가지 범주로 분류됩니다. 약결합 HPC 사례는 전체 시뮬레이션에서 다중 또는 병렬 프로세스의 상호 작용이 강하지 않은 경우를 말합니다. 강결합 HPC 사례는 병렬 프로세스가 동시에 실행되고 시뮬레이션의 각 반복 또는 단계에서 주기적으로 정보를 상호 교환하는 경우를 말합니다.

약결합 워크로드에서 전체 계산 또는 시뮬레이션을 완료하려면 수백에서 수백만 건의 병렬 프로세스가 필요합니다. 이러한 프로세스는 시뮬레이션이 진행되는 동안 임의의 속도에서 임의의 순서로 발생합니다. 따라서 약결합 시뮬레이션의 경우 필요한 컴퓨팅 인프라를 유연하게 선택할 수 있습니다.

강결합 워크로드에는 시뮬레이션의 각 반복에서 주기적으로 정보를 교환하는 프로세스가 포함됩니다. 일반적으로 이러한 강결합 시뮬레이션은 동종 클러스터에서 실행됩니다. 총 코어 또는 프로세스 수는 수십에서 수천 범위일 수 있으며 인프라가 허용한다면 수십만에 달할 수 있습니다. 시뮬레이션 중에 프로세스의 상호 작용이 발생하기 때문에 컴퓨팅 노드 및 네트워크 인프라 같은 인프라에 대한 수요가 증가합니다.

아주 다양한 약결합 및 강결합 애플리케이션을 실행하는 데 사용되는 인프라는 노드 전체의 프로세스 상호 작용에 대한 능력으로 구분됩니다. 기본적인 측면은 두 시나리오 모두에 적용되지만 각각에는 특정 설계 고려 사항이 있습니다. AWS 에서 HPC 인프라를 선택할 때는 두 시나리오에 적용되는 다음과 같은 기본적인 측면을 고려하십시오.

- **네트워크**: 네트워크 요구 사항은 요구 사항이 적은 사례(예: 통신 트래픽이 최소한인 약결합 애플리케이션)부터 대용량 대역폭 및 짧은 지연 시간의 고성능 네트워크를 필요로 하는 강결합 대규모 병렬 애플리케이션까지 다양할 수 있습니다.

Archived

- **스토리지**: HPC 계산은 고유한 방식으로 데이터를 사용하고 생성하고 이동합니다. 스토리지 인프라는 계산의 각 단계에서 이러한 요구 사항을 지원해야 합니다. 시작할 때는 입력 데이터가 주로 저장되고, 실행 중에는 추가 데이터가 생성되고 저장되며, 실행이 완료되면 출력 데이터가 저장소 위치로 이동합니다. 고려해야 할 요인으로는 데이터 크기, 미디어 유형, 전송 속도, 공유 액세스 및 스토리지 속성(예: 내구성 및 가용성)이 있습니다. 노드 간에 공유 파일 시스템을 사용하면 도움이 됩니다. 예를 들어 Amazon Elastic File System(EFS) 같은 NFS(네트워크 파일 시스템) 공유 또는 Amazon FSx for Lustre 같은 Lustre 파일 시스템을 사용합니다.
- **컴퓨팅**: Amazon EC2 인스턴스 유형은 HPC 워크로드에 사용할 수 있는 하드웨어 기능을 정의합니다. 하드웨어 기능으로는 프로세서 유형, 코어 주파수, 프로세서 기능(예: 벡터 확장), 메모리 대 코어 비율 및 네트워크 성능이 있습니다. AWS 에서 인스턴스는 HPC 노드와 같은 것으로 간주됩니다. 이러한 용어는 이 백서에서 상호 교환적으로 사용됩니다.
    - AWS 는 기반 EC2 인스턴스 유형을 선택할 필요 없이 컴퓨팅에 액세스할 수 있는 기능을 갖춘 관리형 서비스를 제공합니다. AWS Lambda 및 AWS Fargate 는 기반 서버를 프로비저닝하고 관리할 필요 없이 워크로드를 실행할 수 있는 컴퓨팅 서비스입니다.
- **배포**: AWS 는 HPC 워크로드 배포를 위한 다수의 옵션을 제공합니다. AWS Management Console 에서는 인스턴스를 수동으로 시작할 수 있습니다. 자동 배포의 경우 다양한 SDK(소프트웨어 개발 키트)를사용하여 다양한 프로그래밍 언어로 완벽한 솔루션을 코딩할 수 있습니다. 가장 많이 사용되는 HPC 배포 옵션 중 하나는 AWS CLI(AWS 명령줄 인터페이스)에서 Bash 셸 스크립팅을 사용하는 것입니다.
    - AWS CloudFormation 템플릿을 사용하면 애플리케이션에 맞춰진 HPC 클러스터의 사양을 코드로 설명하여 몇 분 안에 시작할 수 있습니다. AWS ParallelCluster 는 CloudFormation 과 기존 클러스터 환경의 이미 설치된 소프트웨어(예: 컴파일러 및 스케줄러)를 사용하여 클러스터 시작을 조정하는 오픈 소스 소프트웨어입니다.
    - AWS 는 Amazon EC2 Container Service(Amazon ECS), Amazon Elastic Kubernetes Service(Amazon EKS), AWS Fargate 및 AWS Batch 등 컨테이너 기반 워크로드를 위한 관리형 배포 서비스를 제공합니다.
    - AWS Marketplace 와 APN(AWS 파트너 네트워크)에서 타사의 추가 소프트웨어 옵션을 사용할 수 있습니다.

Archived

클라우드 컴퓨팅을 사용하면 인프라 구성 요소 및 아키텍처 설계를 실험하기가 쉽습니다. 인스턴스 유형, EBS 볼륨 유형, 배포 방법 등을 테스트하여 가장 저렴한 비용으로 최상의 성능을 제공하는 조합을 찾아보십시오.

약결합 시나리오

약결합 워크로드는 다수의 작은 작업을 처리합니다. 일반적으로 작은 작업은 단일 프로세스 또는 다중 프로세스를 사용하여 단일 노드에서 실행되고 해당 노드 내의 병렬 처리를 위해 SMP(Shared Memory Parallelization)가 사용됩니다.

병렬 프로세스 또는 시뮬레이션의 반복은 후처리를 통해 시뮬레이션에서 단일 솔루션 또는 검색을 형성합니다. 몬테카를로 시뮬레이션, 이미지 처리, 유전체학 분석 및 EDA(전자 설계 자동화) 같은 많은 영역에서 약결합 애플리케이션을 찾아볼 수 있습니다.

약결합 워크로드에서는 단일 노드 또는 작업이 손실되더라도 전체 계산이 지연되지 않습니다. 손실된 작업을 나중에 선택하거나 함께 생략할 수 있습니다. 계산에 참여한 노드는 사양 및 성능에 따라 달라질 수 있습니다.

약결합 워크로드에 적합한 아키텍처에 대한 고려 사항은 다음과 같습니다.

- **네트워크**: 병렬 처리에서 일반적으로 상호 작용이 발생하지 않으므로 워크로드의 실행 가능성 또는 성능이 인스턴스 간 네트워크의 대역폭 및 지연 시간 기능에 민감하지 않습니다. 따라서 이 시나리오에는 클러스터링된 배치 그룹이 필요하지 않습니다. 복원력을 저해할 뿐 아니라 성능 이점도 제공하지 않기 때문입니다.
- **스토리지**: 약결합 워크로드의 스토리지 요구 사항은 각기 다르며 데이터 전송, 읽기 및 쓰기에 대한 데이터 세트 크기와 원하는 성능에 따라 결정됩니다.
- **컴퓨팅**: 각 애플리케이션은 다르지만 일반적으로 애플리케이션의 메모리 대 컴퓨팅 비율에 따라 기반 EC2 인스턴스 유형이 결정됩니다. 일부 애플리케이션은 EC2 인스턴스의 GPU(그래픽 처리 장치) 또는 FPGA(필드 프로그래머블 게이트 어레이) 액셀러레이터를 활용하도록 최적화됩니다.
- **배포**: 약결합 시뮬레이션은 종종 가용 영역 전체에 상주할 수 있는 다수(간혹 수백만 개)의 컴퓨팅 코어에서 성능 저하 없이 실행됩니다. 약결합 시뮬레이션은 AWS Batch 및 AWS ParallelCluster 같은 엔드투엔드 서비스 및 솔루션을 통해 배포되거나 Amazon Simple Queue Service(Amazon SQS), Auto Scaling, AWS Lambda 및 AWS Step Functions 같은 AWS 의 조합을 통해 배포될 수 있습니다.

Archived

강결합 시나리오

강결합 애플리케이션은 상호 종속성을 기반으로 계산을 실행하는 병렬 프로세스로 구성됩니다. 약결합 계산과 달리 강결합 시뮬레이션의 모든 프로세스는 함께 반복되므로 상호 통신이 필요합니다. 반복은 전체 시뮬레이션의 한 단계로 정의됩니다. 강결합 계산은 수만 개의 프로세스 또는 코어를 통해 1 회에서 수백만 회 이상 반복됩니다. 노드 1 개에 장애가 발생하면 전체 계산이 실패하는 것이 일반적입니다. 전체 실패의 위험을 완화하기 위해, 알려진 상태에서 시뮬레이션 다시 시작할 수 있게 하는 애플리케이션 수준 체크포인트 지정이 계산 중에 주기적으로 수행됩니다.

이러한 시뮬레이션은 프로세스 간 통신에 MPI(메시지 전달 인터페이스)를 사용합니다. OpenMP 를 통한 공유 메모리 병렬 처리를 MPI 에서 사용할 수 있습니다. 강결합 HPC 워크로드의 예로는 컴퓨터 유체 역학, 기상 예측 및 저류 시뮬레이션이 있습니다.

강결합 HPC 워크로드에 적합한 아키텍처에 대한 고려 사항은 다음과 같습니다.

- **네트워크**: 강결합 계산의 네트워크 요구 사항은 까다롭습니다. 노드 간 통신 속도가 느리면 전체 계산이 느려집니다. 네트워킹 성능을 일관적으로 유지하려면 가장 큰 인스턴스 크기, 향상된 네트워킹 및 클러스터 배치 그룹이 필요합니다. 이러한 기술은 시뮬레이션 실행 시간을 최소화하고 전체 비용을 줄여줍니다. 강결합 애플리케이션의 크기는 다양합니다. 단일의 큰 문제는 다수의 프로세스 또는 코어에 걸쳐 배치되며 일반적으로 병렬화됩니다. 총 계산 요구 사항이 낮은 다수의 작은 사례는 네트워크 요구 사항에 가장 큰 부담을 줍니다. 특정 Amazon EC2 인스턴스는 EFA(Elastic Fabric Adapter)를 네트워크 인터페이스로 사용하여 높은 수준의 노드 간 통신이 요구되는 애플리케이션을 AWS 에서 대규모로 실행합니다. EFA 의 맞춤형 운영 체제 바이패스 하드웨어 인터페이스는 강결합 애플리케이션의 크기 조정에 중요한 역할을 하는 인스턴스 간 통신을 개선합니다.
- **스토리지**: 강결합 워크로드의 스토리지 요구 사항은 각기 다르며 데이터 전송, 읽기 및 쓰기에 대한 데이터 세트 크기와 원하는 성능에 따라 결정됩니다. 임시 데이터 스토리지 또는 스크래치 공간을 특별히 고려해야 합니다.
- **컴퓨팅**: EC2 인스턴스는 코어 대 메모리 비율이 각기 다른 다양한 구성으로 제공됩니다. 병렬 애플리케이션의 경우 메모리 집약적인 병렬 시뮬레이션을 추가 컴퓨팅 노드에 분산해 코어당 메모리 요구 사항을 낮추고 최상의 성능을 제공하는 인스턴스 유형을 대상으로 지정하는 것이 도움이 됩니다. 강결합 애플리케이션에는 유사한 컴퓨팅 노드에서 구축된 동종 클러스터가 필요합니다. 가장 큰 인스턴스 크기를 대상으로

Archived

지정하면 노드 간 네트워크 지연 시간을 최소화하는 동시에 노드 간 통신을 위한 최대한의 네트워크 성능을 제공할 수 있습니다.

- **배포**: 다양한 배포 옵션을 사용할 수 있습니다. "기존" 클러스터 환경에서 시뮬레이션을 시작할 때와 마찬가지로 엔드투엔드 자동화를 달성할 수 있습니다. 클라우드 확장성을 활용하여 수백 개의 대규모 다중 프로세스 사례를 한 번에 실행하여 대기열의 필요성을 제거할 수 있습니다. 강결합 시뮬레이션은 AWS Batch 및 AWS ParallelCluster 같은 엔드투엔드 솔루션을 사용하거나 CloudFormation 또는 EC2 플릿 같은 AWS 서비스 기반 솔루션을 통해 배포될 수 있습니다.

참조 아키텍처

약결합 및 강결합 워크로드에는 많은 아키텍처가 적용되며 시나리오에 따라 약간의 수정이 필요할 수 있습니다. 기존의 온프레미스 클러스터는 클러스터 인프라에 획일적인 접근 방식을 강제합니다. 그러나 클라우드는 방대한 가능성을 제공하며 성능 및 비용 최적화를 가능하게 합니다. 클라우드에서는 스케줄러 및 로그인 노드가 포함된 기존의 클러스터 환경부터 클라우드 네이티브 솔루션이 제공하는 비용 효율성의 장점을 갖춘 클라우드 네이티브 아키텍처에 이르는 다양한 구성을 활용할 수 있습니다. 아래에 참조 아키텍처 5 개가 나와 있습니다.

1. 기존 클러스터 환경
2. Batch 기반 아키텍처
3. 대기열 기반 아키텍처
4. 하이브리드 배포
5. 서버리스 워크플로

기존 클러스터 환경

많은 사용자가 기존의 HPC 환경과 유사한 환경에서 클라우드 여정을 시작합니다. 이 환경은 로그인 노드와 작업 실행을 위한 스케줄러로 구성되는 경우가 많습니다.

일반적으로 기존 클러스터를 프로비저닝할 때는 AWS CloudFormation 템플릿을 사용하여 컴퓨팅 클러스터를 프로비저닝하고 사용자의 특정 작업에 맞게 사용자 지정하는 방법이 사용됩니다. AWS ParallelCluster 는 AWS CloudFormation 에 기반을 둔 엔드투엔드 클러스터

Archived

프로비저닝 기능의 한 예입니다. 템플릿 안에는 아키텍처에 대한 복잡한 설명이 숨겨져 있지만 사용자는 일반적인 구성 옵션을 통해 인스턴스 유형, 스케줄러 또는 부트스트랩 작업(예: 애플리케이션 설치 또는 데이터 동기화)을 선택할 수 있습니다. 이 템플릿을 구성하고 실행하면 기존의 HPC 클러스터의 "형태 및 모양"과 유사하지만 확장성이라는 이점이 추가된 HPC 환경을 제공할 수 있습니다. 로그인 노드에는 스케줄러, 공유 파일 시스템 및 실행 환경이 유지됩니다. 또한 자동 크기 조정 메커니즘을 사용하면 작업이 작업 대기열에 제출될 때 추가 인스턴스를 구동할 수 있습니다. 유휴 상태로 전환된 인스턴스는 자동으로 종료됩니다.

클러스터를 영구 구성으로 배포하거나 임시 리소스로 사용할 수 있습니다. 영구 클러스터는 로그인 인스턴스 및 컴퓨팅 플릿을 사용하여 배포됩니다. 컴퓨팅 플릿은 고정된 크기이거나 제출된 작업의 수에 따라 컴퓨팅 플릿의 수를 늘리고 줄이는 Auto Scaling 그룹에 연결될 수 있습니다. 영구 클러스터에는 항상 실행 중인 인프라가 있습니다. 이와 달리 클러스터를 임시 리소스로 사용하여 각 워크로드를 자체 클러스터에서 실행할 수도 있습니다. 임시 클러스터는 자동화를 통해 활성화됩니다. 예를 들어 AWS CLI 에서 Bash 스크립트를 사용하거나 AWS SDK 와 함께 Python 스크립트를 사용하여 엔드투엔드 사례 자동화를 제공할 수 있습니다. 사례가 제출될 때마다 리소스가 프로비저닝 및 시작되고, 데이터가 노드에 배치되며, 작업이 여러 노드에서 실행되고, 사례 출력이 자동으로 검색되거나 Amazon S3 로 전송됩니다. 작업이 완료되면 인프라가 종료됩니다. 이러한 클러스터는 인프라를 코드로 처리하고, 비용을 최적화하며, 완전한 버전 제어를 통해 인프라 변경 사항을 추적합니다.

기존 클러스터 아키텍처는 약결합 워크로드와 강결합 워크로드에 모두 사용될 수 있습니다. 최상의 성능을 원한다면 강결합 워크로드에 동종 인스턴스 유형으로 클러스터링된 배치 그룹의 컴퓨팅 플릿을 사용해야 합니다.

Archived

참조 아키텍처

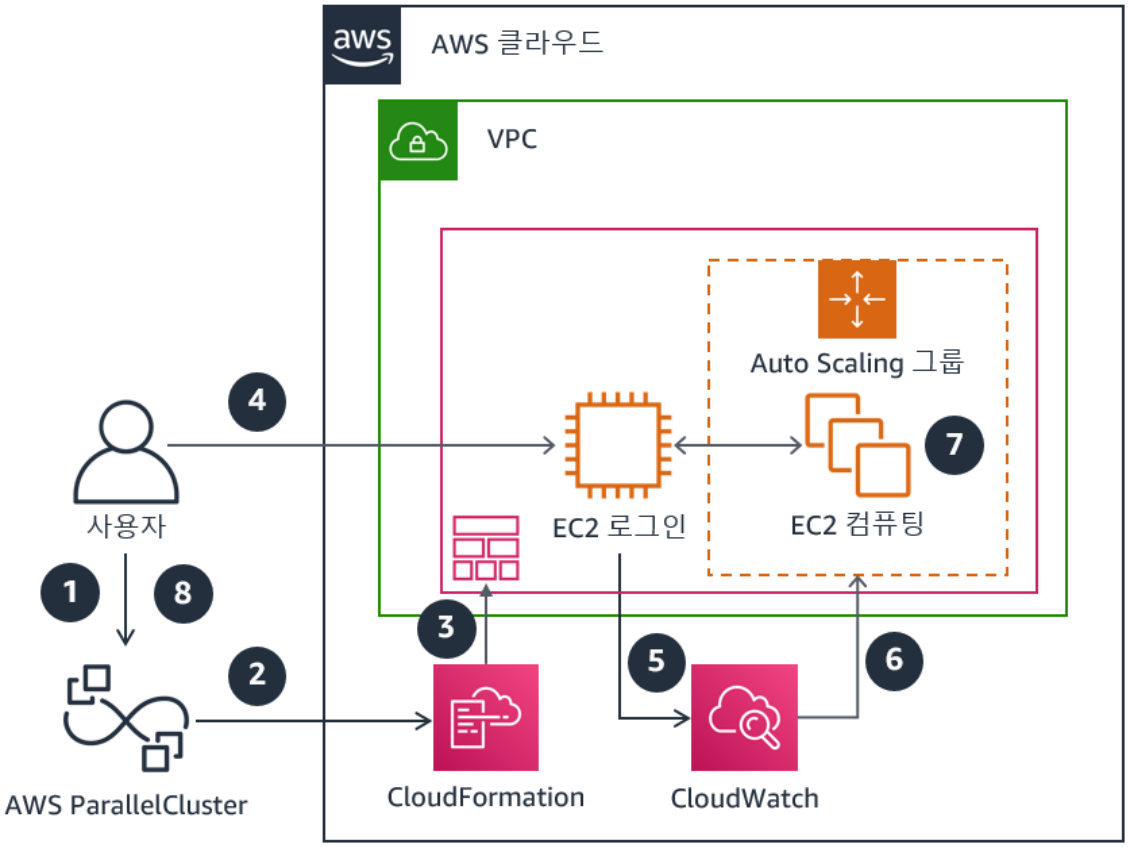

그림 1: AWS ParallelCluster 로 배포된 기존 클러스터

워크플로 단계:

1. 사용자가 AWS ParallelCluster CLI 와 구성 파일의 사양을 사용하여 클러스터 생성을 시작합니다.
2. AWS CloudFormation 이 일부 사용자 지정 설정(예: 구성 파일 편집 또는 웹 인터페이스 사용)이 지정된 클러스터 템플릿 파일에 설명된 대로 클러스터 아키텍처를 구축합니다.
3. AWS CloudFormation 이 사용자 지정된 HPC 소프트웨어/애플리케이션을 통해 생성된 EBS 스냅샷에서 인프라를 배포합니다. 클러스터 인스턴스에서는 NFS 내보내기를 통해 인프라에 액세스할 수 있습니다.
4. 사용자가 로그인 인스턴스에 로그인하고 스케줄러(예: SGE, Slurm)에 작업을 제출합니다.
5. 로그인 인스턴스가 작업 대기열 크기에 따라 CloudWatch 로 지표를 내보냅니다.

6. 작업 대기열 크기가 임계값을 초과하는 경우 CloudWatch 가 컴퓨팅 인스턴스의 수를 늘리는 Auto Scaling 이벤트를 트리거합니다.
7. 예약된 작업이 컴퓨팅 플릿을 통해 처리됩니다.
8. [선택 사항] 사용자가 클러스터 삭제 및 모든 리소스의 종료를 시작합니다.

Batch 기반 아키텍처

AWS Batch 는 리소스 프로비저닝 또는 스케줄러 관리 없이 대규모 컴퓨팅 워크로드를 실행할 수 있는 완전관리형 서비스입니다.[3] AWS Batch 를 사용하면 개발자, 과학자 및 엔지니어가 AWS 에서 수많은 배치 컴퓨팅 작업을 효율적으로 손쉽게 실행할 수 있습니다. AWS Batch 는 제출된 배치 작업의 볼륨 및 지정된 리소스 요구 사항에 따라 최적의 수량 및 유형의 컴퓨팅 리소스(예: CPU 또는 메모리 최적화 인스턴스)를 동적으로 프로비저닝합니다. AWS Batch 는 Amazon EC2[4]와 스팟 인스턴스 등 AWS 컴퓨팅 서비스 및 기능의 전 범위에 걸쳐 배치 컴퓨팅 워크로드를 계획, 예약, 실행합니다.[5] 작업 실행에 필요한 배치 컴퓨팅 소프트웨어 또는 서버 클러스터를 설치하고 관리할 필요가 없으므로 결과를 분석하고 새로운 통찰력을 얻는 데 집중할 수 있습니다.

AWS Batch 를 사용하는 경우 애플리케이션을 컨테이너로 패키징하고, 작업의 종속성을 지정한 다음, AWS Management Console, CLI 또는 SDK 를 사용하여 배치 작업을 제출하면 됩니다. 실행 파라미터 및 작업 종속성을 지정하고 다양한 주요 배치 컴퓨팅 워크플로 엔진 및 언어(예: Pegasus WMS, Luigi 및 AWS Step Functions)와 통합할 수 있습니다. AWS Batch 가 제공하는 기본 작업 대기열 및 컴퓨팅 환경 정의를 사용하여 빠르게 시작할 수 있습니다.

AWS Batch 기반 아키텍처는 약결합 워크로드와 강결합 워크로드에 모두 사용될 수 있습니다. 강결합 워크로드에는 AWS Batch 에서 다중 노드 병렬 작업을 사용해야 합니다.

Archived

참조 아키텍처

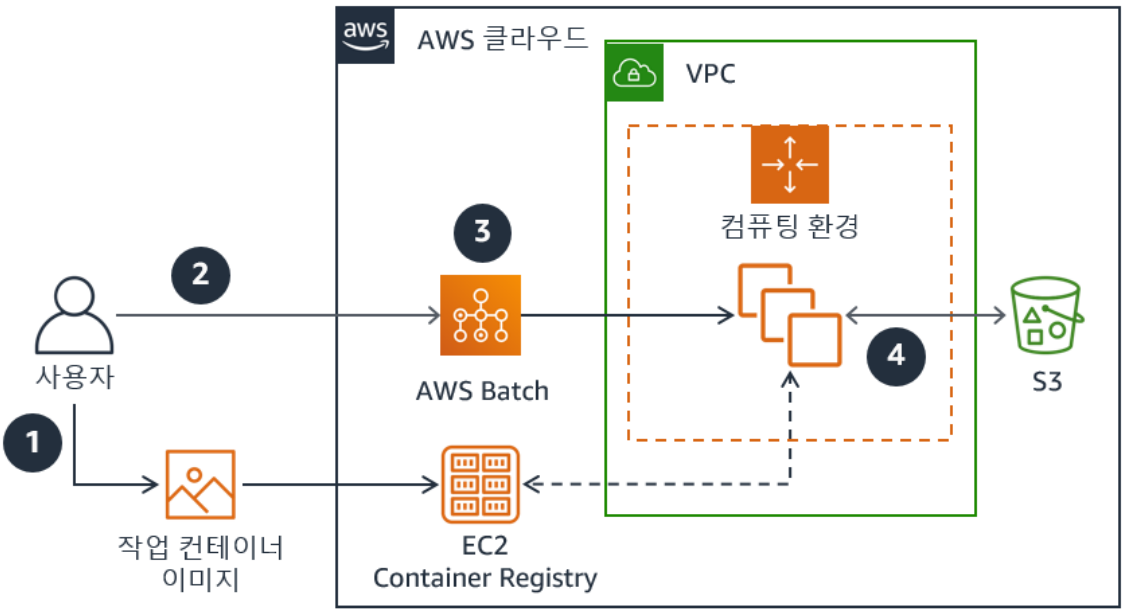

그림 2: AWS Batch 아키텍처 예

워크플로 단계:

1. 사용자가 작업 컨테이너를 생성하고, 컨테이너를 Amazon EC2 Container Registry(Amazon ECR) 또는 다른 컨테이너 레지스트리(예: DockerHub)에 업로드한 후 AWS Batch 에 작업 정의를 생성합니다.
2. 사용자가 AWS Batch 의 작업 대기열에 작업을 제출합니다.
3. AWS Batch 가 컨테이너 레지스트리에서 이미지를 가져오고 대기열의 작업을 처리합니다.
4. 각 작업의 입력 및 출력 데이터가 S3 버킷에 저장됩니다.

대기열 기반 아키텍처

Amazon SQS 는 완전관리형 메시지 대기열 서비스입니다. 이 서비스를 사용하면 전처리 단계를 컴퓨팅 단계 및 후처리 단계에서 손쉽게 분리할 수 있습니다.[6] 고유한 기능을 수행하는 개별 구성 요소에서 애플리케이션을 구축하면 확장성과 안정성이 개선됩니다. 구성 요소 분리는 최신 애플리케이션 설계의 모범 사례 중 하나입니다. Amazon SQS 는 클라우드 네이티브 약결합 솔루션의 핵심 솔루션입니다.

Archived

Amazon SQS 는 AWS CLI 또는 AWS SDK 스크립팅 솔루션과 함께 사용되는 경우가 많은데 이렇게 하면 사용자가 AWS 구성 요소와 직접 상호 작용하지 않고 데스크톱에서 애플리케이션을 배포할 수 있습니다. AWS Batch 같은 서비스 관리형 배포와 달리 SQS 및 EC2 가 포함된 대기열 기반 아키텍처에는 자체 관리형 컴퓨팅 인프라가 필요합니다.

대기열 기반 아키텍처는 약결합 워크로드에 가장 적합하며 강결합 워크로드에 적용할 경우 금방 복잡해질 수 있습니다.

참조 아키텍처

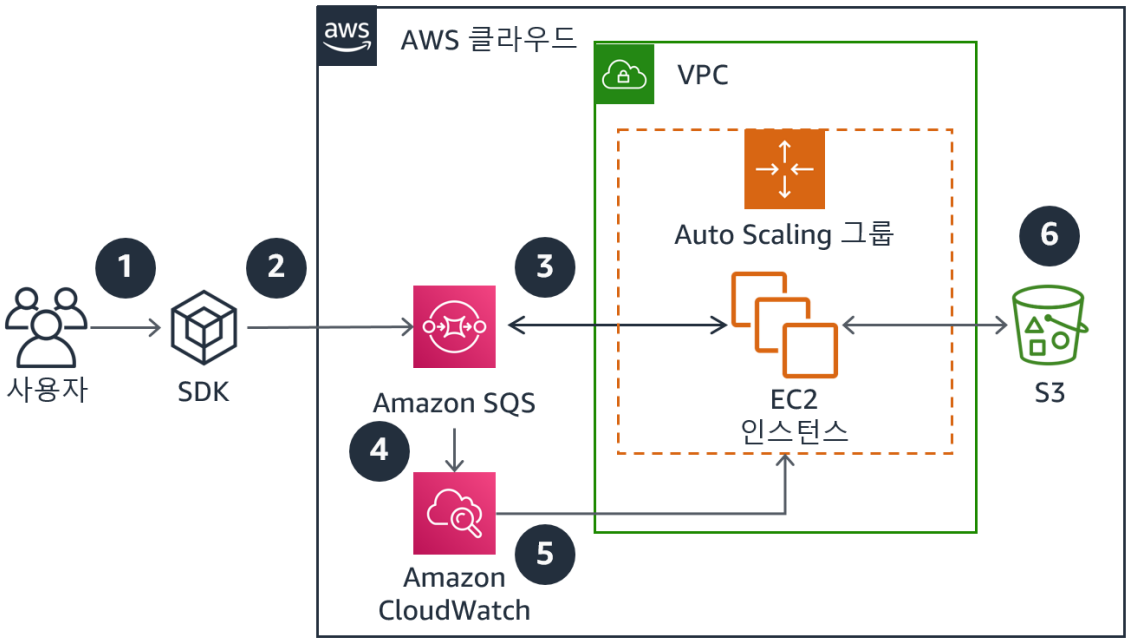

그림 3: 약결합 워크로드를 위해 배포된 Amazon SQS

워크플로 단계:

1. 여러 사용자가 AWS CLI 또는 SDK 를 통해 작업을 제출합니다.
2. 작업은 Amazon SQS 대기열에 메시지 형태로 배치됩니다.
3. EC2 인스턴스가 대기열을 폴링하고 작업 처리를 시작합니다.
4. Amazon SQS 가 대기열의 메시지(작업) 수에 따라 지표를 내보냅니다.
5. 대기열이 지정된 길이보다 길어질 경우 Auto Scaling 에 알림을 전송하는 Amazon CloudWatch 경보가 구성됩니다. Auto Scaling 이 EC2 인스턴스 수를 늘립니다.
6. EC2 인스턴스가 소스 데이터를 가져와서 S3 버킷에 결과 데이터를 저장합니다.

하이브리드 배포

하이브리드 배포는 온프레미스 인프라에 투자한 조직이 AWS 도 사용하려는 경우 주로 고려하는 배포 방법입니다. 이 접근 방식에서는 조직이 온프레미스 리소스를 보강하고, 즉각적인 전체 마이그레이션 대신 대체 AWS 마이그레이션 경로를 생성할 수 있습니다.

하이브리드 시나리오는 워크로드 분리 같은 최소한의 조정부터 스케줄러 기반 작업 배치와 같이 긴밀하게 통합된 접근 방식에 이르기까지 다양합니다. 예를 들어 어떤 조직에서는 워크로드를 분리한 후 특정 유형의 모든 워크로드를 AWS 인프라에서 실행할 수 있습니다. 온프레미스 프로세스 및 인프라에 많은 투자를 한 조직이라면 기존의 작업 스케줄링 소프트웨어 및 작업 제출 포털을 통해 AWS 리소스를 관리하여 최종 사용자에게 보다 원활한 경험을 제공하고자 할 수 있습니다. AWS 리소스를 필요에 따라 동적으로 프로비저닝하고 프로비저닝 해제할 수 있는 기능을 제공하는 작업 스케줄러(상용 및 오픈 소스)가 많습니다. 기반 리소스 관리에는 네이티브 AWS 통합(예: AWS CLI 또는 API)이 사용되며 스케줄러에 따라 고도로 사용자 지정된 환경을 구성하는 것이 가능할 수 있습니다. 작업 스케줄러는 AWS 리소스 관리에 도움이 되지만 스케줄러는 성공적인 배포의 한 가지 측면일 뿐입니다.

하이브리드 시나리오를 성공적으로 운영하는 데 있어서 중요한 요소는 데이터 위치 및 데이터 이동입니다. 중요한 데이터 세트가 필요하지 않거나 이러한 데이터 세트를 생성하지 않는 일부 HPC 워크로드의 경우 데이터 관리가 그다지 중요하지 않습니다. 그러나 대량의 입력 데이터가 필요하거나 중요한 출력 데이터를 생성하는 작업은 병목 현상이 될 수 있습니다. 데이터 관리를 해결하는 기술은 조직에 따라 다릅니다. 예를 들어 최종 사용자가 작업 제출 스크립트로 데이터 전송을 관리하는 조직이 있고, 데이터 세트가 상주하는 위치에서 특정 작업만 실행하는 조직이 있으며, 두 위치의 데이터를 복제하도록 선택하는 조직이 있는가 하면, 여러 옵션을 조합하여 사용하도록 선택하는 조직도 있습니다.

데이터 관리 접근 방식에 따라 AWS 는 하이브리드 배포를 지원하는 다양한 서비스를 제공합니다. 예를 들어 AWS Direct Connect 는 온프레미스 환경과 AWS 간의 전용 네트워크 연결을 설정하고, AWS DataSync 는 온프레미스 스토리지의 데이터를 Amazon S3 또는 Amazon Elastic File System 으로 자동으로 이동합니다. AWS Marketplace 와 APN(AWS 파트너 네트워크)에서 타사의 추가 소프트웨어 옵션을 사용할 수 있습니다.

하이브리드 배포 아키텍처는 약결합 워크로드와 강결합 워크로드에 모두 사용될 수 있습니다. 그러나 단일의 강결합 워크로드는 온프레미스 또는 AWS 에 배치되어야 최상의 성능을 실현할 수 있습니다.

Archived

참조 아키텍처

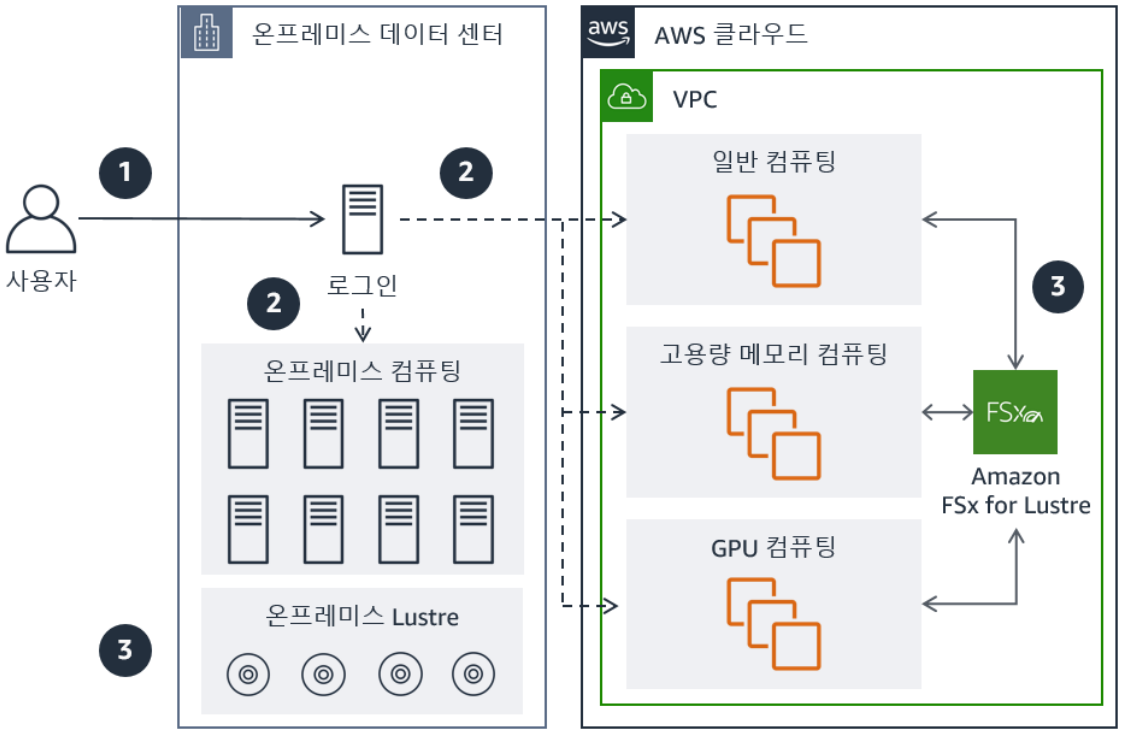

그림 3: 하이브리드 스케줄러 기반 배포 예

워크플로 단계:

1. 사용자가 온프레미스 로그인 노드의 스케줄러(예: Slurm)에 작업을 제출합니다.
2. 스케줄러는 구성에 따라 온프레미스 컴퓨팅 또는 AWS 인프라에서 작업을 실행합니다.
3. 작업은 실행 위치에 따라 공유 스토리지에 액세스합니다.

서버리스

약결합 클라우드 여정은 완전히 서버리스인 환경으로 연결되는 경우가 많습니다. 즉, 사용자는 애플리케이션에만 집중하고 서버 프로비저닝 책임을 관리형 서비스로 넘길 수 있습니다. AWS Lambda 를 사용하면 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다. 사용한 컴퓨팅 시간에 대해서만 요금을 지불하면 되고 코드가 실행되지 않을 때는 요금이 부과되지 않습니다. 코드를 업로드하면 코드를 실행하고 크기를 조정하는 데 필요한 모든 작업을 Lambda 가 처리합니다. 또한 Lambda 에는 다른 AWS 서비스의 이벤트를 자동으로 트리거하는 기능도 있습니다.

서버리스 Lambda 접근 방식의 두 번째 장점은 확장성입니다. 각 작업자의 크기는 메모리가 장착된 컴퓨팅 코어와 같이 보통일 수 있지만 아키텍처에서 수천 개의 동시 Lambda 작업자를 생성할 수 있으므로 HPC 레이블에 걸맞는 대용량 컴퓨팅 처리 용량을 제공할 수 있습니다. 예를 들어 동일한 알고리즘을 호출하여 많은 수의 파일을 분석하거나, 다수의 유전체를 동시에 분석하거나, 단일 유전체 내의 많은 유전자 사이트를 모델링할 수 있습니다. 크기 조정에서 실현 가능한 최대 규모와 그 속도가 중요합니다. 서버 기반 아키텍처의 경우 요청에 대한 응답으로 용량을 늘리려면 EC2 인스턴스 같은 가상 머신을 사용하는 경우에도 몇 분의 시간이 필요하지만 서버리스 Lambda 함수는 몇 초 안에 확장됩니다. AWS Lambda 를 사용하면 컴퓨팅 집약적 결과에 대해 예측되지 않은 모든 요청에 즉시 응답하고, 리소스를 미리 프로비저닝하여 낭비할 필요 없이 요청 수의 변동을 처리할 수 있는 HPC 인프라를 구성할 수 있습니다.

컴퓨팅 외에 HPC 워크플로를 지원하는 다른 서버리스 아키텍처도 있습니다. AWS Step Functions 를 사용하면 여러 AWS 서비스를 하나로 결합하여 파이프라인의 여러 단계를 조정할 수 있습니다. 예를 들어 AWS Step Functions 를 조정에 사용하고, Amazon S3 를 스토리지로 사용하며, AWS Lambda 를 소규모 작업에 사용하고, AWS Batch 를 데이터 처리에 사용하는 방법으로 자동화된 유전체학 파이프라인을 생성할 수 있습니다.

서버리스 아키텍처는 약결합 워크로드에 가장 적합하며 다른 HPC 아키텍처와 함께 사용하는 경우 워크플로 조정을 수행하기에 적합합니다.

Archived

참조 아키텍처

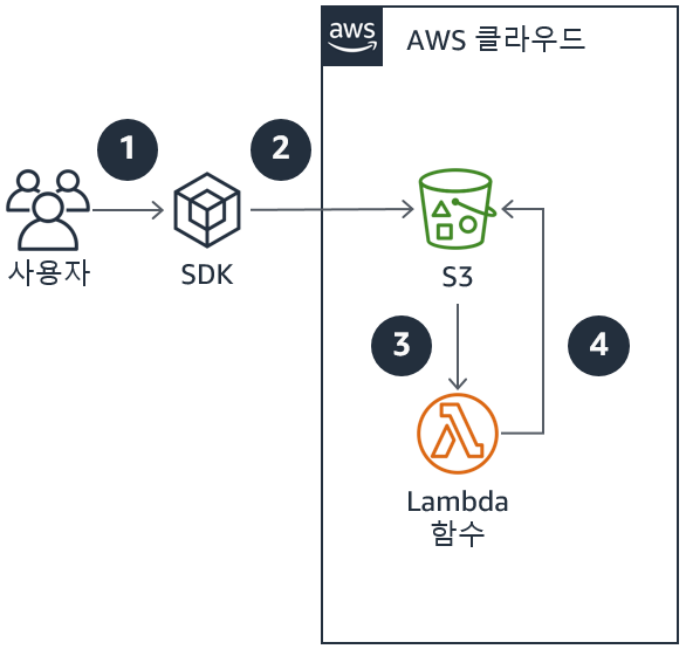

그림 4: Lambda 로 배포된 약결합 워크로드 예

워크플로 단계:

1. 사용자가 AWS CLI 또는 SDK 를 통해 S3 버킷에 파일을 업로드합니다.
2. 입력 파일이 수신 접두사(예: input/)와 함께 저장됩니다.
3. S3 이벤트가 수신 데이터의 처리를 위한 Lambda 함수를 자동으로 트리거합니다.
4. 출력 파일이 발신 접두사(예: output/)와 함께 다시 S3 버킷에 저장됩니다.

Archived

Well-Architected 프레임워크의 5 가지 기반

이 섹션에서는 Well-Architected 프레임워크의 5 가지 기반을 중심으로 HPC 를 설명합니다. 각 기반은 설계 원칙, 정의, 모범 사례, 평가 질문, 고려 사항, 주요 AWS 서비스 및 유용한 링크를 설명합니다.

운영 우수성 기반

운영 우수성 기반에는 비즈니스 가치를 실현하고 지원 프로세스 및 절차를 지속적으로 개선하기 위해 시스템을 실행하고 모니터링하는 기능이 포함됩니다.

설계 원칙

클라우드에서는 다수의 원칙에 따라 운영 우수성을 달성할 수 있습니다. 다음은 특히 HPC 워크로드에 대해 강조되는 원칙입니다. AWS Well-Architected 프레임워크 백서의 설계 원칙도 참조하십시오.

- **클러스터 작업 자동화**: 클라우드에서는 전체 워크로드를 코드로 정의하고 코드를 사용하여 업데이트할 수 있습니다. 코드를 사용하면 반복적인 프로세스 또는 절차를 자동화할 수 있습니다. 자동화의 이점은 일관적으로 인프라를 재현하고 운영 절차를 구현할 수 있다는 것입니다. 예를 들어 작업 제출 프로세스 및 이벤트(예: 작업 시작, 완료 또는 실패) 응답을 자동화할 수 있습니다. HPC 에서는 모든 작업에서 사용자가 사례 파일을 업로드하고, 스케줄러에 작업을 제출하고, 결과 파일을 이동하는 등의 여러 단계를 반복하는 것이 일반적입니다. 이 반복적인 단계를 스크립트 또는 이벤트 기반 코드로 자동화하면 사용 편의성을 극대화하고 비용 및 오류를 최소화할 수 있습니다.
- **가능한 경우 클라우드 네이티브 아키텍처 사용**: HPC 아키텍처에는 일반적으로 2 가지 형태가 있습니다. 첫 번째는 기존 클러스터 구성과 로그인 인스턴스, 컴퓨팅 노드 및 작업 스케줄러가 포함되는 형태의 아키텍처입니다. 두 번째는 자동화된 배포 및 관리형 서비스를 사용하는 클라우드 네이티브 아키텍처입니다. 단일 워크로드를 각 (임시) 클러스터에 대해 실행하거나 서버리스 기능을 사용할 수 있습니다. 클라우드 네이티브 아키텍처에서는 고급 기술을 민주화하여 작업을 최적화할 수 있지만 가장 좋은 기술 접근 방식은 HPC 사용자가 원하는 환경과 일치합니다.

Archived

정의

클라우드의 운영 우수성에는 3 가지 모범 사례 영역이 있습니다.

- 준비
- 운영
- 개선

준비, **운영** 및 **개선** 영역에 대한 자세한 내용은 AWS Well-Architected 프레임워크 백서를 참조하십시오. 개선 영역은 이 백서에서 설명하지 않습니다.

모범 사례

준비

AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

워크로드 배포를 준비할 때는 전문 소프트웨어 패키지(상용 또는 오픈 소스)를 사용하여 시스템 정보를 파악하고 이 정보를 활용하여 워크로드에 대한 아키텍처 패턴을 정의하는 것이 좋습니다. AWS ParallelCluster 또는 AWS CloudFormation 같은 자동화 도구를 사용하면 변수로 구성 가능한 방식으로 이러한 아키텍처를 정의할 수 있습니다.

클라우드는 다수의 스케줄링 옵션을 제공합니다. 한 가지 옵션은 단일 노드 작업과 다중 노드 작업을 모두 지원하는 완전관리형 배치 처리 서비스인 AWS Batch 를 사용하는 것입니다. 다른 옵션은 스케줄러를 사용하지 않는 것입니다. 예를 들어 임시 클러스터를 생성하여 단일 작업을 직접 실행할 수 있습니다.

> HPCOPS 1: 클러스터 전체의 아키텍처를 표준화할 때 사용하는 방법은 무엇입니까?
>
> HPCOPS 2: 작업을 예약할 때 사용하는 방법(기존 스케줄러, AWS Batch 또는 스케줄러가 없는 임시 클러스터)은 무엇입니까?

Archived

운영

운영을 표준화하고 정기적으로 관리해야 합니다. 자동화를 활용하고, 소규모로 자주 변경하며, 정기적으로 품질 보증 테스트를 수행하고, 변경의 추적, 감사, 롤백 및 검토를 위한 메커니즘을 정의해야 합니다. 대규모의 드문 변경, 다운타임을 예약해야 하는 변경 및 수동 실행이 필요한 변경을 방지하십시오. 워크로드의 주요 운영 지표를 바탕으로 다양한 로그 및 지표를 수집하고 검토하여 무중단 운영을 보장해야 합니다.

AWS 는 HPC 작업을 처리할 수 있는 추가 도구를 제공합니다. 모니터링 지원부터 배포 자동화에 이르는 다양한 도구를 사용할 수 있습니다. 예를 들어 Auto Scaling 을 사용하여 실패한 인스턴스를 다시 시작하거나, CloudWatch 로 클러스터의 로드 지표를 모니터링하거나, 작업이 완료되는 시기에 대한 알림을 구성하거나, 관리형 서비스(예: AWS Batch)를 사용하여 실패한 작업에 대한 재시도 규칙을 구현할 수 있습니다. 클라우드 네이티브 도구를 사용하면 애플리케이션 배포 및 변경 관리가 크게 개선될 수 있습니다.

소규모의 증분 변경을 기준으로 릴리스 관리 프로세스(수동 또는 자동)를 수행하고 버전을 추적해야 합니다. 릴리스에 문제가 있는 경우 운영에 미치는 영향 없이 릴리스를 되돌릴 수 있어야 합니다. AWS CodePipeline 및 AWS CodeDeploy 같은 연속 통합 및 연속 배포 도구를 사용하여 변경 배포를 자동화하십시오. AWS CodeCommit 같은 버전 제어 도구와 AWS CloudFormation 템플릿 같은 인프라 구성 및 자동화 도구를 사용하여 소스 코드 변경을 추적합니다.

> HPCOPS 3: 워크로드를 변경할 때 변경의 영향을 최소화하기 위해 어떤 방법을 사용하고 있습니까?
>
> HPCOPS 4: 워크로드를 모니터링하여 워크로드가 예상대로 작동하는지 확인하는 방법은 무엇입니까?

HPC 에 클라우드를 사용하는 경우 운영 측면에서 새로운 고려 사항이 있습니다. 온프레미스 클러스터는 크기가 고정되지만 클라우드 클러스터는 수요에 맞게 확장될 수 있습니다. 또한 HPC 의 클라우드 네이티브 아키텍처는 온프레미스 아키텍처와 다른 방식으로 작동합니다. 예를 들어 작업 제출 메커니즘과 작업이 도착할 때 온디맨드 인스턴스 리소스를 프로비저닝하는 메커니즘이 온프레미스와 다릅니다. 따라서 클라우드의 탄력성과 클라우드 네이티브 아키텍처의 동적 특성을 수용하는 운영 절차를 채택해야 합니다.

Archived

개선

개선 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. 자세한 내용은 AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 참조하십시오.

보안 기반

보안 기반에는 정보, 시스템, 자산을 보호하는 동시에 위험 진단과 문제 해결 전략을 통해 비즈니스 가치를 제공하는 기능이 포함됩니다.

설계 원칙

클라우드에서는 다양한 원칙에 따라 시스템 보안을 강화할 수 있습니다. AWS Well-Architected 프레임워크 백서의 설계 원칙이 권장되며 HPC 워크로드에도 마찬가지로 적용됩니다.

정의

클라우드의 보안에는 5 가지 모범 사례 영역이 있습니다.

- IAM(자격 증명 및 액세스 관리)
- 탐지 제어
- 인프라 보호
- 데이터 보호
- 인시던트 대응

시스템 아키텍처를 설계하기 전에 보안 사례를 설정해야 합니다. 권한을 제어하고, 보안 인시던트를 식별하며, 시스템 및 서비스를 보호하고, 데이터 보호를 통해 데이터의 기밀성 및 무결성을 유지해야 합니다. 보안 인시던트 대응을 위한 프로세스를 올바르게 정의하고 시행해야 합니다. 데이터 손실을 방지하고 규제 의무를 준수하는 등의 목표를 지원하려면 이러한 도구 및 기술이 중요합니다.

AWS 의 공동 책임 모델은 클라우드 채택 조직에서 보안 및 규정 준수 목표를 달성하는 데 도움이 됩니다. 클라우드 서비스를 지원하는 인프라를 AWS 가 물리적으로 보호하기 때문에 조직에서는 서비스를 이용하여 목표를 달성하는 데 집중할 수 있습니다. 또한 AWS 클라우드에서 보안 데이터에 액세스하고 자동화된 방식으로 보안 이벤트에 대응할 수 있습니다.

Archived

모든 보안 모범 사례 영역은 중요하며 AWS Well-Architected 프레임워크 백서에 문서화되어 있습니다. **탐지 제어**, **인프라 보호** 및 **인시던트 대응** 영역은 AWS Well-Architected 프레임워크 백서에 설명되어 있으므로 이 백서에서는 설명하지 않으며 HPC 워크로드에 대해 수정이 필요하지 않습니다.

모범 사례

IAM(자격 증명 및 액세스 관리)

자격 증명 및 액세스 관리는 정보 보안 프로그램에서 중요한 부분입니다. IAM 은 권한이 있는 인증된 사용자만 리소스에 액세스할 수 있도록 하는 것입니다. 예를 들어 보안 주체(계정에서 작업을 수행하는 사용자, 그룹, 서비스 및 역할)를 정의하고, 이러한 보안 주체를 참조하는 정의된 정책을 구축하고, 강력한 자격 증명 관리를 구현합니다. 이러한 권한 관리 요소는 인증 및 권한 부여의 핵심 개념을 형성합니다.

HPC 워크로드를 자율적이고 일시적인 방식으로 실행하면 중요한 데이터의 노출을 제한할 수 있습니다. 자율 배포를 사용하면 인스턴스에 대한 인적 액세스가 최소화되므로 리소스 노출 또한 최소화됩니다. HPC 데이터가 제한된 시간 안에 생성되므로 잠재적 무단 데이터 액세스의 가능성이 최소화됩니다.

> HPCSEC 1: 워크로드 인프라에 대한 인적 액세스를 최소화하기 위해 관리형 서비스, 자율 방법 및 임시 클러스터를 어떻게 사용하고 있습니까?

HPC 아키텍처에는 다양한 관리형(예: AWS Batch, AWS Lambda) 및 비관리형 컴퓨팅 서비스(예: Amazon EC2)가 사용될 수 있습니다. 아키텍처에서 컴퓨팅 환경에 직접 액세스해야 하는 경우(예: EC2 인스턴스에 연결) 사용자는 일반적으로 SSH(Secure Shell)를 통해 연결하고 SSH 키를 사용하여 인증합니다. 이 액세스 모델은 기존 클러스터 시나리오에서 일반적입니다. SSH 키를 포함한 모든 자격 증명은 적절히 보호되고 정기적으로 교체되어야 합니다.

또는 AWS Systems Manager 의 완전관리형 서비스(Session Manager)를 통해 대화하여 브라우저 기반 셸 및 CLI 환경을 사용할 수도 있습니다. 이 서비스는 인바운드 포트를 열고 배스천 호스트를 유지하고 SSH 키를 관리할 필요 없이 안전하고 감사 가능한 인스턴스 관리를 제공합니다. Session Manager 에는 ProxyCommand 를 지원하는 모든 SSH 클라이언트를 통해 액세스할 수 있습니다.

Archived

HPCSEC 2: 자격 증명을 보호하고 관리하기 위해 어떤 방법을 사용하고 있습니까?

탐지 제어

탐지 제어 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

인프라 보호

인프라 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

데이터 보호

시스템 아키텍처를 설계하기 전에 기본적인 보안 사례를 설정해야 합니다. 예를 들어 데이터 분류는 민감도에 따라 조직의 데이터를 구분하는 방법이고, 암호화는 무단 액세스 시 데이터 해석을 불가능하게 만들어 데이터를 보호하는 방법입니다. 이러한 도구 및 기술은 데이터 손실을 방지하거나 규제 의무를 준수하는 등의 목표를 지원한다는 점에서 중요합니다.

HPCSEC 3: 아키텍처에서 결과의 수명 주기 동안 스토리지 가용성 및 내구성에 대한 데이터 요구 사항을 어떻게 해결합니까?

민감도 및 규제 의무 외에 데이터의 후속 사용 시기 및 방법을 기준으로 HPC 데이터를 분류할 수도 있습니다. 최종 결과는 보존되는 경우가 많지만 중간 결과는 필요한 경우 다시 생성할 수 있으므로 보존하지 않아도 될 수 있습니다. 데이터를 신중하게 평가하고 분류하면 중요한 데이터를 프로그래밍 방식으로 복원력이 더 좋은 스토리지 솔루션(예: Amazon S3 및 Amazon EFS)으로 마이그레이션할 수 있습니다.

데이터 지속성과 데이터를 프로그래밍 방식으로 처리하는 방법을 이해하면 Well-Architected 인프라의 노출을 최소화하고 보호를 극대화할 수 있습니다.

인시던트 대응

인시던트 대응 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

Archived

안정성 기반

안정성 기반에는 인프라 또는 서비스 중단으로부터 시스템을 복구하고, 수요를 충족할 컴퓨팅 리소스를 동적으로 확보하며, 잘못된 구성 또는 일시적 네트워크 문제와 같은 중단을 완화하는 기능이 포함됩니다.

설계 원칙

클라우드에서는 다수의 원칙을 적용하여 안정성을 개선할 수 있습니다. 다음은 특히 HPC 워크로드에 대해 강조되는 원칙입니다. 자세한 내용은 AWS Well-Architected 프레임워크 백서의 설계 원칙을 참조하십시오.

- **수평적 크기 조정으로 시스템 전체 가용성 증대**: 수평적 크기 조정 옵션을 사용하면 단일 장애가 전체 시스템에 미치는 영향을 줄일 수 있습니다. 예를 들어 단일의 대규모 공유 HPC 클러스터에서 다수의 작업을 실행하는 것보다 Amazon 인프라에 여러 클러스터를 생성하여 잠재적 장애 위험을 추가로 격리하는 것이 좋습니다. 인프라를 코드로 처리할 수 있으므로 단일 클러스터 안에서 리소스 크기를 수평적으로 조정하고 개별 사례를 실행하는 다수의 클러스터 크기를 수평적으로 조정할 수 있습니다.
- **용량 추정 불필요**: HPC 클러스터 세트 1 개를 프로비저닝하여 현재 요구 사항을 충족하고 수동 또는 자동 크기 조정을 통해 수요 증가 또는 감소를 처리할 수 있습니다. 예를 들어 사용되지 않는 유휴 컴퓨터 노드를 종료합니다. 예를 들어 대기열에서 대기하는 것보다 사용되지 않는 유휴 컴퓨터 노드를 종료하고 동시 클러스터를 실행하여 여러 계산을 처리하는 것이 좋습니다.
- **자동화에서 변경 관리**: 인프라 변경을 자동화하면 클러스터 인프라의 버전을 제어하고 이전에 생성된 클러스터를 정확히 복제할 수 있습니다. 자동화 변경은 관리가 필요합니다.

정의

클라우드의 안정성에는 3 가지 모범 사례 영역이 있습니다.

- 기반
- 변경 관리
- 오류 관리

변경 관리 영역은 AWS Well-Architected 프레임워크 백서에 설명되어 있습니다.

Archived

모범 사례

기반

HPCREL 1: 보유 계정에 대한 AWS Service Limits 를 어떻게 관리하고 있습니까?

AWS Service Limits(팀에서 요청할 수 있는 각 리소스 수의 상한)는 실수로 인한 리소스 오버프로비저닝을 방지하기 위해 설정됩니다.

HPC 애플리케이션에는 많은 수의 컴퓨팅 인스턴스가 동시에 사용됩니다. 수평적 크기 조정 기능은 HPC 워크로드에 많은 이점을 제공합니다. 그러나 수평적 크기 조정을 통해 대규모 워크로드를 단일의 대규모 클러스터 또는 다수의 소규모 클러스터에 한 번에 배포하려면 먼저 AWS Service Limits 를 늘려야 할 수 있습니다.

많은 경우 대규모 배포의 요구 사항을 처리하려면 Service Limits 를 기본값에서 높여야 합니다. 증량을 요청하려면 AWS Support 에 문의하십시오.

변경 관리

변경 관리 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

오류 관리

모든 복잡한 시스템에서는 가끔 오류가 발생할 수 있습니다. 그러므로 이러한 오류에 대해 알고, 대응하고, 재발을 방지하는 것이 중요합니다. 오류 시나리오에는 클러스터 시작 오류 또는 특정 워크로드 오류가 포함될 수 있습니다.

HPCREL 2: 애플리케이션을 장애에서 복구할 때 체크포인트를 어떻게 사용합니까?

여러 방법으로 내결함성을 개선할 수 있습니다. 장기 실행 사례의 경우 정기적 체크포인트를 코드에 포함하면 장애 발생 시 부분적인 상태에서 작업을 계속할 수 있습니다. 체크포인트는 많은 HPC 애플리케이션에 이미 포함되어 있는 애플리케이션 수준 오류 관리의 일반적인 기능입니다. 가장 일반적인 접근 방식은 애플리케이션에서 중간 결과를 주기적으로 쓰도록 하는 것입니다. 중간 결과를 기록하면 애플리케이션 오류를 파악할 수 있는 가능성이 있고 작업의 일부를 잃더라도 필요에 따라 사례를 다시 시작할 수 있습니다.

Archived

체크포인트는 고도로 경제적이지만 중단될 수 있는 인스턴스를 사용하는 스팟 인스턴스에 유용합니다. 또한 일부 애플리케이션의 경우 기본 스팟 중단 동작을 변경(예: 인스턴스를 종료하는 대신 중지하거나 최대 절전 모드로 전환)하는 것이 도움이 될 수 있습니다. 체크포인트를 오류 관리에 사용하는 경우 스토리지 옵션의 내구력을 고려하는 것이 중요합니다.

HPCREL 3: 아키텍처의 내결함성을 어떻게 계획했습니까?

여러 가용 영역에 배포하면 내결함성을 개선할 수 있습니다. 강결합 HPC 애플리케이션에는 짧은 지연 시간이 요구됩니다. 따라서 각 개별 사례가 단일 클러스터 배치 그룹과 가용 영역 내에 상주해야 합니다. 반대로, 약결합 애플리케이션의 경우 이러한 짧은 지연 시간이 요구되지 않으므로 여러 가용 영역에 배포하는 기능을 통해 오류 관리를 개선할 수 있습니다.

이 설계 의사 결정을 내릴 때는 안정성 기반과 비용 기반을 절충하십시오. 컴퓨팅 및 스토리지 인프라가 중복(예: 헤드 노드와 연결된 스토리지)되면 추가 비용이 발생하며 데이터를 가용 영역 또는 다른 AWS 리전으로 이동할 때 데이터 전송 요금이 발생할 수 있습니다. 긴급하지 않은 사용 사례의 경우 DR(재해 복구) 이벤트 중에만 다른 가용 영역으로 이동하는 것이 좋습니다.

성능 효율성 기반

성능 효율성 기반에서는 컴퓨팅 리소스를 효율적으로 사용하여 요구 사항을 충족하고 수요 변경 및 기술 변화에 따라 이러한 효율성을 유지하는 데 중점을 둡니다.

설계 원칙

클라우드에서 HPC 를 설계하는 경우 다양한 원칙에 따라 성능 효율성을 높일 수 있습니다.

- **애플리케이션에 적합한 클러스터 설계**: 기존 클러스터는 정적이기 때문에 클러스터에 맞춰 애플리케이션을 설계해야 합니다. AWS 에서는 애플리케이션에 맞춰 클러스터를 설계할 수 있습니다. 각 애플리케이션에 개별 클러스터를 사용할 수 있으므로 획일적인 모델이 더 이상 필요하지 않습니다. AWS 에서 다양한 애플리케이션을 실행하는 경우 다양한 아키텍처를 사용하여 각 애플리케이션의 요구 사항을 충족할 수 있습니다. 이렇게 하면 최소한의 비용으로 최상의 성능을 얻을 수 있습니다.
- **유의미한 사용 사례를 사용하여 성능 테스트**: 특정 아키텍처에서 HPC 애플리케이션의 성능을 측정하는 가장 좋은 방법은 애플리케이션 자체의 유의미한 데모를 실행하는

Archived

것입니다. 예상 컴퓨팅, 메모리, 데이터 전송 또는 네트워크 트래픽 없이 너무 작거나 큰 데모 사례를 사용할 경우 AWS 에서 애플리케이션 성능을 제대로 테스트할 수 없습니다. 시스템별 벤치마크를 통해 기반 컴퓨팅 인프라 성능을 파악할 수도 있지만 이러한 벤치마크에는 애플리케이션의 집계 성능이 반영되지 않습니다. AWS 는 종량과금제 모델을 제공하므로 빠르고 경제적으로 개념 증명을 수행할 수 있습니다.

- **가능한 경우 클라우드 네이티브 아키텍처 사용**: 클라우드에서 관리형, 서버리스 및 클라우드 네이티브 아키텍처를 사용하면 서버를 실행하고 유지 관리할 필요 없이 기존의 컴퓨팅 작업을 수행할 수 있습니다. HPC 의 클라우드 네이티브 구성 요소는 컴퓨팅, 스토리지, 작업 오케스트레이션과 데이터 및 메타데이터 구성 기능을 제공합니다. 다양한 AWS 서비스를 사용하여 워크로드 프로세스의 각 단계를 분리하고 성능 개선에 도움이 되는 기능에 맞춰 최적화할 수 있습니다.
- **실험 횟수 증가**: 가상 리소스 및 자동화 가능한 리소스를 사용하여 다양한 유형의 인스턴스, 스토리지 및 구성으로 비교 테스트를 수행할 수 있습니다.

정의

클라우드의 성능 효율성에는 4 가지 모범 사례 영역이 있습니다.

- 선택
- 검토
- 모니터링
- 절충

검토, **모니터링** 및 **절충** 영역은 AWS Well-Architected 프레임워크 백서에 설명되어 있습니다.

모범 사례

선택

특정 시스템에 대한 최적의 솔루션은 워크로드 종류에 따라 다릅니다. Well-Architected 시스템은 다수의 솔루션을 사용하며 서로 다른 기능으로 성능을 개선할 수 있습니다. HPC 아키텍처에는 대기열, 배치, 클러스터 컴퓨팅, 컨테이너, 서버리스 및 이벤트 기반 등 하나 이상의 다른 아키텍처 요소가 사용될 수 있습니다.

Archived

컴퓨팅

HPCPERF 1: 컴퓨팅 솔루션을 어떻게 선택합니까?

특정 HPC 아키텍처에 대한 최적의 컴퓨팅 솔루션은 워크로드 배포 방법, 자동화 수준, 사용량 패턴 및 구성에 따라 다릅니다. 프로세스 각 단계에 대해 서로 다른 컴퓨팅 솔루션을 선택하여 사용 수 있습니다. 아키텍처에 대해 잘못된 컴퓨팅 솔루션을 선택하면 성능 효율성이 저하될 수 있습니다.

인스턴스는 가상화된 서버이며 서로 다른 패밀리 및 크기로 제공되어 다양한 기능을 제공합니다. 일부 인스턴스 패밀리는 컴퓨팅, 메모리 또는 GPU 집약적인 워크로드와 같은 특정 워크로드를 대상으로 합니다. 범용 인스턴스도 있습니다.

워크로드 특정 인스턴스 패밀리와 범용 인스턴스 패밀리 모두 HPC 애플리케이션에 유용하게 사용됩니다. HPC 와 특히 관련된 인스턴스로는 컴퓨팅 최적화 패밀리 및 가속화된 인스턴스 유형(GPU 및 FPGA)이 있습니다.

일부 인스턴스 패밀리는 패밀리 안에서 추가 기능을 위한 변형을 제공합니다. 예를 들어 로컬 스토리지, 향상된 네트워크 기능 또는 다른 종류의 프로세서를 제공하는 변형이 인스턴스 패밀리에 포함될 수 있습니다. 인스턴스 유형 매트릭스[7]에서 일부 HPC 워크로드의 성능을 개선하는 이러한 변형을 확인할 수 있습니다.

각 인스턴스 패밀리에 포함된 하나 이상의 인스턴스 크기를 사용하여 리소스 크기를 수직으로 조정할 수 있습니다. 애플리케이션이 지원하는 프로세스 수에 따라 어떤 애플리케이션에는 더 큰 인스턴스 유형(예: 24xlarge)이 필요하고, 다른 애플리케이션은 이보다 작은 유형(예: large)에서 실행됩니다. 강결합 워크로드로 작업을 수행하는 경우 가장 큰 인스턴스 유형에서 최적의 성능을 얻을 수 있습니다.

T 시리즈 인스턴스 패밀리는 CPU 사용량이 중간이고 CPU 성능의 기준 수준 이상으로 버스트할 경우 성능 이점이 있는 애플리케이션을 위해 설계되었습니다. 대부분의 HPC 애플리케이션은 컴퓨팅 집약적이기 때문에 T 시리즈 인스턴스 패밀리를 사용하는 경우 성능 저하가 발생합니다.

애플리케이션의 요구 사항(예: 원하는 코어 수, 프로세서 속도, 메모리 요구 사항, 스토리지 요구 사항 및 네트워킹 시양)은 서로 다릅니다. 인스턴스 패밀리 및 유형을 선택할 때는 애플리케이션의 특정 요구 사항을 먼저 파악해야 합니다. 특정 애플리케이션 구성 요소에 지정된 인스턴스를 사용해야 하는 애플리케이션의 경우 인스턴스 유형을 혼합하여 사용할 수 있습니다.

컨테이너는 운영 체제를 가상화하는 한 가지 방법으로, 많은 HPC 워크로드에 주로 사용되며 애플리케이션이 이미 컨테이너화된 경우에 특히 적합합니다. AWS Batch, Amazon Elastic

Archived

Container Service(ECS) 및 Amazon Elastic Container Service for Kubernetes(EKS) 같은 AWS 서비스는 컨테이너화된 애플리케이션을 배포하는 데 유용합니다.

함수는 실행 환경을 추상화합니다. AWS Lambda 를 사용하면 인스턴스의 배포, 실행 또는 유지 관리 없이 코드를 실행할 수 있습니다. 많은 AWS 서비스에서 서비스 내부 활동에 따라 이벤트를 내보내는데 서비스 이벤트에서 Lambda 함수를 트리거할 수 있는 경우가 많습니다. 예를 들어 객체를 Amazon S3 에 업로드한 후 Lambda 함수를 실행할 수 있습니다. 많은 HPC 사용자가 Lambda 를 사용하여 워크플로 중에 코드를 자동으로 실행합니다.

선택한 컴퓨팅 인스턴스를 시작할 때는 여러 옵션을 선택해야 합니다.

- **운영 체제**: 최상의 성능을 달성하고 최신 라이브러리에 대한 액세스를 보장하려면 최신 운영 체제를 사용하는 것이 중요합니다.
- **가상화 유형**: AWS Nitro System 에서는 최신 세대의 EC2 인스턴스 유형이 실행됩니다. Nitro System 은 모든 호스트 하드웨어의 컴퓨팅 및 메모리 리소스를 인스턴스에 제공하므로 성능이 전반적으로 개선됩니다. 전용 니트로 카드를 사용하여 고속 네트워킹, 고속 EBS 및 I/O 가속화를 지원할 수 있습니다. 인스턴스에서 관리 소프트웨어를 위한 리소스를 따로 유지할 필요가 없습니다.

니트로 하이퍼바이저는 메모리 및 CPU 할당을 관리하고 베어 메탈과 구별되지 않는 성능을 제공하는 경량 하이퍼바이저입니다. Nitro System 에서 니트로 하이퍼바이저 없이 베어 메탈 인스턴스를 실행할 수도 있습니다. 베어 메탈 인스턴스를 시작하면 기반 서버가 부스트되고 모든 하드웨어 및 펌웨어 구성 요소가 확인됩니다. 따라서 베어 메탈 인스턴스를 사용하여 워크로드를 시작하려면 가상화된 인스턴스를 사용할 때보다 시간이 더 걸릴 수 있습니다. 온디맨드로 리소스가 시작되고 종료되는 동적 HPC 환경을 운영하는 경우에는 시작 시간이 추가된다는 점을 고려해야 합니다.

> HPCPERF 2: 애플리케이션에 맞게 컴퓨팅 환경을 최적화할 때 어떤 방법을 사용하고 있습니까?

기반 하드웨어 기능: AMI 를 선택하는 것 외에 기반 인텔 프로세서의 하드웨어 기능을 활용하여 환경을 추가로 최적화할 수 있습니다. 기반 하드웨어를 최적화할 때는 다음과 같은 4 가지 기본적인 방법을 고려할 수 있습니다.

Archived

1. 고급 프로세서 기능
2. 인텔 하이퍼스레딩 기술
3. 프로세서 선호도
4. 프로세서 상태 제어

고급 프로세서 기능(예: 고급 벡터 확장)은 HPC 애플리케이션에 유용하며 인텔 아키텍처에 맞게 소프트웨어를 컴파일링할 경우 계산 속도가 개선될 수 있습니다.[8] 아키텍처별 명령에 대한 컴파일러 옵션은 컴파일러에 따라 다릅니다(컴파일러 사용 설명서 참조).

AWS 는 인텔의 하이퍼스레딩 기술(일반적으로 "하이퍼스레딩")을 기본적으로 지원합니다. 일부 애플리케이션의 경우 하이퍼스레딩을 통해 하이퍼스레드당 프로세스 1 개(코어당 프로세스 2 개)를 사용하여 성능을 개선할 수 있습니다. 대부분의 HPC 애플리케이션은 하이퍼스레딩을 비활성화하는 것이 유리하므로 HPC 애플리케이션의 환경에서는 기본적으로 하이퍼스레딩이 비활성화되는 경향이 있습니다. 하이퍼스레딩은 Amazon EC2 에서 간편하게 비활성화됩니다. 하이퍼스레딩을 활성화한 상태에서 애플리케이션을 테스트한 경우가 아니라면 하이퍼스레딩을 비활성화하고 HPC 애플리케이션을 실행할 때 프로세스를 시작하고 개별적으로 코어에 고정하는 것이 좋습니다. CPU 또는 프로세서 선호도를 사용하면 프로세스를 손쉽게 고정할 수 있습니다.

프로세서 선호도는 다양한 방법으로 제어가 가능합니다. 예를 들어 운영 체제 수준(Windows 및 Linux)에서 구성하거나, 스레딩 라이브러리 안에 컴파일러 플래그로 설정하거나, 실행 중에 MPI 플래그로 프로세서 선호도를 지정할 수 있습니다. 프로세서 선호도 제어에 대해 선택하는 방법은 워크로드 및 애플리케이션에 따라 다릅니다.

AWS 에서는 특정 인스턴스 유형에 대한 프로세서 상태 제어를 튜닝할 수 있습니다.[9] C 상태(유휴 상태) 및 P 상태(작동 상태) 설정을 변경하여 성능을 최적화할 수 있습니다. 대부분의 워크로드에서는 기본 C 상태 및 P 상태 설정으로 최적의 성능을 얻을 수 있습니다. 그러나 단일 또는 이중 코어 주파수를 높여 지연 시간을 낮추거나 터보 부스트 스파이크 주파수와 반대로 주파수를 낮춰 일관적인 성능을 얻는 것이 애플리케이션에 유리하다면 일부 인스턴스에서 사용 가능한 C 상태 또는 P 상태 설정으로 최적화를 실험해 보십시오.

컴퓨팅 환경을 최적화할 때 사용할 수 있는 컴퓨팅 옵션은 많습니다. 클라우드 배포에서는 운영 체제부터 인스턴스 유형과 베어 메탈 배포에 이르는 모든 수준에서 최적화를 실험할 수 있습니다. 정적 클러스터의 경우 배포 전에 튜닝이 수행되므로 원하는 성능을 달성하려면 클라우드 기반 클러스터로 실험하는 것이 좋습니다.

Archived

스토리지

HPCPERF 3: 스토리지 솔루션을 어떻게 선택합니까?

특정 HPC 아키텍처에 대한 최적의 스토리지 솔루션은 대부분 해당 아키텍처의 대상이 되는 개별 애플리케이션에 따라 다릅니다. 워크로드 배포 방법, 자동화 수준 및 원하는 데이터 수명 주기 패턴도 고려해야 할 요인입니다. AWS 는 다양한 스토리지 옵션을 제공합니다. 컴퓨팅과 마찬가지로, 애플리케이션의 특정 스토리지 요구 사항을 해결하는 것을 목표로 해야 최상의 성능을 실현할 수 있습니다. AWS 에서는 "획일적인" 접근 방식으로 스토리지를 오버프로비저닝할 필요가 없으며 대규모의 고속 공유 파일 시스템을 항상 사용해야 하는 것도 아닙니다. HPC 성능을 최적화하려면 컴퓨팅 선택을 최적화하는 것이 중요하며 스토리지 솔루션의 속도는 대부분의 HPC 애플리케이션에 그다지 영향을 미치지 않습니다.

HPC 배포에는 클러스터 컴퓨팅 노드에서 액세스하는 공유 또는 고성능 파일 시스템이 필요한 경우가 많습니다. AWS Managed Services, AWS Marketplace 오퍼링, APN 파트너 솔루션 및 EC2 인스턴스에 배포된 오픈 소스 구성에서 다수의 아키텍처 패턴을 사용하여 이러한 스토리지 솔루션을 구현할 수 있습니다. 구체적으로, Amazon FSx for Lustre 는 고성능 병렬 파일 시스템이 필요한 HPC 아키텍처에 경제적인 고성능 솔루션을 제공하는 관리형 서비스입니다. 또한 Amazon EBS 볼륨 또는 인스턴스 스토어 볼륨이 장착된 EC2 인스턴스 또는 Amazon Elastic File System(EFS)에서는 공유 파일 시스템을 생성할 수 있습니다. 공유 디렉터리를 생성할 때는 단순한 NFS 마운트가 주로 사용됩니다.

스토리지 솔루션을 선택할 때 EBS 지원 인스턴스를 선택하여 로컬 스토리지 또는 공유 스토리지로 사용하거나 둘 다로 사용할 수 있습니다. EBS 볼륨은 HPC 스토리지 솔루션의 기준으로 사용되는 경우가 많습니다. 마그네틱 HDD(하드 디스크 드라이브), 범용 SSD(Solid State Drive) 및 높은 IOPS 솔루션을 제공하는 프로비저닝된 IOPS SSD 를 포함하여 다양한 유형의 EBS 볼륨을 사용할 수 있습니다. 이러한 솔루션은 처리량, IOPS 성능 및 비용이 각각 다릅니다.

Amazon EBS 최적화 인스턴스를 선택할 경우 성능을 추가로 개선할 수 있습니다. 최적화된 구성 스택을 사용하는 EBS 최적화 인스턴스는 Amazon EBS I/O 전용 용량을 추가로 제공합니다. 이 최적화를 수행하면 Amazon EBS I/O 와 인스턴스의 다른 네트워크 트래픽 간의 경합이 최소화되므로 EBS 볼륨의 성능을 최대한 높일 수 있습니다. 일관적인 성능을 원하는 경우와 HPC 애플리케이션에서 지연 시간이 짧은 네트워크를 사용하거나 I/O 집약적인 데이터를 EBS 볼륨에 저장해야 하는 경우 EBS 최적화 인스턴스를 선택하십시오.

Archived

EBS 최적화 인스턴스를 시작하려면 기본적으로 EBS 최적화를 지원하는 인스턴스 유형을 선택하거나 시작 시 EBS 최적화를 활성화할 수 있는 인스턴스 유형을 선택합니다.

NVMe(Nonvolatile Memory express) SSD 볼륨(특정 인스턴스 패밀리에서만 사용 가능)과 같은 인스턴스 스토어 볼륨을 임시 블록 수준 스토리지로 사용할 수 있습니다. EBS 최적화 및 인스턴스 스토어 볼륨 지원은 인스턴스 유형 매트릭스를 참조하십시오.[10]

스토리지 솔루션을 선택할 때는 원하는 성능을 달성하는 데 필요한 액세스 패턴과 일치하는지 확인하십시오. 서로 다른 스토리지 유형 및 구성을 사용하여 손쉽게 실험할 수 있습니다. HPC 워크로드의 경우 가장 비싼 옵션이 항상 최고의 성능을 제공하는 것은 아닙니다.

네트워킹

> HPCPERF 4: 네트워크 솔루션을 어떻게 선택합니까?

HPC 워크로드에 대한 최적의 네트워크 솔루션은 지연 시간, 대역폭 및 처리량 요구 사항에 따라 다릅니다. 강결합 HPC 애플리케이션의 경우 컴퓨팅 노드 간의 네트워크 연결 지연 시간이 최대한 짧아야 합니다. 중간 규모의 강결합 워크로드의 경우 다수의 코어가 포함된 큰 인스턴스 유형을 선택하면 네트워크를 전혀 이동하지 않고 해당 인스턴스 내에 애플리케이션을 완전히 배치할 수 있습니다.

네트워크에 바인딩된 일부 애플리케이션에는 고성능 네트워크가 필요합니다. 이러한 애플리케이션의 경우 네트워크 성능이 높은 인스턴스를 선택할 수 있습니다. 패밀리에서 가장 큰 인스턴스 유형이 가장 높은 네트워크 성능을 제공합니다. 자세한 내용은 인스턴스 유형 매트릭스를 참조하십시오.[7]

대규모의 강결합 애플리케이션에는 여러 인스턴스가 필요하며 인스턴스 간 지연 시간이 짧아야 합니다. AWS 에서는 컴퓨팅 노드를 클러스터 배치 그룹(가용 영역 내의 논리적 인스턴스 그룹화)으로 시작하여 이 구성을 구현할 수 있습니다. 클러스터 배치 그룹은 인스턴스 간 완전 이등분 대역폭을 포함하여 차단 및 초과 구독이 없는 연결성을 제공합니다. 여러 인스턴스를 사용하고 지연 시간에 민감한 강결합 애플리케이션에는 클러스터 배치 그룹을 사용하십시오.

강결합 애플리케이션의 경우 클러스터 배치 그룹에 더해 Amazon EC2 인스턴스에 연결되는 네트워크 디바이스인 EFA(Elastic Fabric Adapter)를 사용하는 것도 좋습니다. EFA 는 클라우드 기반 HPC 시스템에 사용되는 기존의 TCP 전송보다 짧고 일관적인 지연 시간과 더 높은 처리량을

Archived

제공합니다. *Libfabric* API 를 통해 OS 바이패스 액세스 모델을 지원하므로 HPC 애플리케이션에서 네트워크 인터페이스 하드웨어와 직접 통신할 수 있습니다. EFA 는 인스턴스 간 통신 성능을 개선하고, 기존 AWS 네트워크 인프라의 작업에 최적화되었으며, 강결합 애플리케이션의 크기 조정에 중요한 역할을 합니다.[13]

애플리케이션에서 EFA 의 OS 바이패스 기능을 사용할 수 없거나 인스턴스 유형이 EFA 를 지원하지 않는 경우에는 향상된 네트워킹을 지원하는 인스턴스 유형을 선택하여 네트워크 성능을 최적화할 수 있습니다. 향상된 네트워킹은 하드웨어 에뮬레이션 디바이스 대신 패스스루를 사용하여 EC2 인스턴스의 네트워킹 성능을 개선하고 CPU 사용률을 낮춥니다. 이 방법을 사용하면 EC2 인스턴스에서 기존의 디바이스 가상화보다 높은 대역폭, 높은 초당 패킷 처리량 및 짧은 인스턴스 간 지연 시간을 달성할 수 있습니다.

향상된 네트워킹은 모든 현재 세대 인스턴스 유형에서 사용할 수 있으며 지원되는 드라이버가 포함된 AMI 가 필요합니다. 최신 AMI 에는 지원되는 드라이버가 포함되지만 사용자 지정 AMI 의 경우 업데이트된 드라이버가 필요할 수 있습니다. 향상된 네트워킹 활성화 및 인스턴스 지원에 대한 자세한 내용은 향상된 네트워킹 설명서를 참조하십시오.[11]

약결합 워크로드의 경우 일반적으로 지연 시간이 매우 짧은 네트워킹이 중요하지 않으며 클러스터 배치 그룹을 사용하거나 동일한 가용 영역 또는 리전에 인스턴스를 유지할 필요가 없습니다.

검토

검토 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

모니터링

모니터링 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

절충

모니터링 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

Archived

비용 최적화 기반

비용 최적화 기반에는 전체 수명 주기 동안 HPC 시스템을 보강하고 개선하는 지속적인 프로세스가 포함됩니다. 최초 개념 증명의 초기 설계부터 지속적인 프로덕션 워크로드 운영에 이르는 전체 과정에 이 백서의 사례를 도입하면 비용을 최소화하면서 원하는 비즈니스 결과를 달성할 수 있는 비용 인식 시스템을 구축하고 운영할 수 있습니다. 이러한 시스템을 사용하면 비즈니스의 투자수익률을 극대화할 수 있습니다.

설계 원칙

클라우드에서는 다수의 원칙에 따라 HPC 비용을 최적화할 수 있습니다.

- **소비 모델 채택**: 사용한 컴퓨팅 리소스에 대한 요금만 지불합니다. HPC 워크로드는 변동이 크기 때문에 필요에 따라 리소스 용량을 늘리고 줄여 비용을 절감할 수 있습니다. 예를 들어 실행률이 낮은 HPC 용량을 사전에 프로비저닝하고 예약하여 할인 혜택을 높이고 버스트 요구 용량을 스팟 또는 온디맨드 요금으로 프로비저닝하여 필요할 때만 온라인으로 전환할 수 있습니다.
- **특정 작업의 인프라 비용 최적화**: HPC 워크로드의 다수는 데이터 전송, 전처리, 컴퓨팅 계산, 후처리, 데이터 전송 및 스토리지 단계를 포함하는 데이터 처리 파이프라인의 일부입니다. 클라우드에서는 고가의 대규모 서버를 사용하는 대신 각 단계에서 컴퓨팅 플랫폼이 최적화됩니다. 예를 들어 파이프라인의 한 단계를 수행하는 데 다량의 메모리가 필요한 경우 메모리 집약적인 애플리케이션을 위한 좀 더 비싼 대용량 메모리 서버를 사용하여 이 사용에 대한 요금만 지불하고, 작고 덜 비싼 컴퓨팅 플랫폼에서 다른 모든 단계를 실행할 수 있습니다. 워크로드의 각 단계에 맞춰 인프라를 최적화하면 비용이 절감됩니다.
- **가장 효율적인 방법으로 워크로드 버스트**: HPC 워크로드의 경우 클라우드에서 수평적 크기 조정을 통해 비용을 절감할 수 있습니다. 수평적으로 크기를 조정하는 경우 전체 워크로드의 많은 작업 또는 반복이 동시에 실행되므로 총 경과 시간이 감소합니다. 애플리케이션에 따라 수평적 크기 조정은 결과를 더 빨리 제공함으로써 간접적으로 비용을 절감하는 중립적 비용 절감 효과를 제공할 수 있습니다.
- **스팟 요금 활용**: Amazon EC2 스팟 인스턴스는 온디맨드 인스턴스에 비해 대폭 할인된 요금으로 AWS 의 예비 컴퓨팅 용량을 제공합니다. 그러나 EC2 에서 용량을 회수해야 하는 경우 스팟 인스턴스가 중단될 수 있습니다. 스팟 인스턴스는 유연하거나 내결함성이

Archived

있는 워크로드에서 가장 경제적으로 사용할 수 있는 리소스입니다. HPC 워크로드는 간헐적인 특성이 있으므로 스팟 인스턴스에 매우 적합합니다. Spot Advisor 를 사용하면 스팟 인스턴스의 중단 위험을 최소화할 수 있습니다. 또한 기본 중단 동작을 변경하고 스팟 플릿을 사용하여 스팟 인스턴스를 관리하면 중단이 미치는 영향을 완화할 수 있습니다. 가끔 워크로드를 다시 시작해야 하지만 이는 스팟 인스턴스가 제공하는 비용 절감으로 쉽게 상쇄됩니다.

- **비용과 시간 사이의 균형 평가**: 강결합의 대규모 병렬 워크로드는 다양한 수의 코어에서 실행될 수 있습니다. 이러한 애플리케이션의 경우 사례의 실행 효율성은 일반적으로 코어 수가 많을수록 떨어집니다. 비용 대 처리 시간 곡선은 유사한 유형과 크기의 많은 사례를 실행할 때 생성될 수 있습니다. 이러한 곡선은 사례 유형 및 애플리케이션과 관련이 있는데 계산 대 네트워크 요구 사항의 비율에 따라 크기 조정이 크게 달라지기 때문입니다. 대규모 워크로드의 경우 소규모 워크로드보다 크기 조정 가능성이 높습니다. 비용과 처리 시간 사이의 균형을 이해하면 시간이 중요한 워크로드를 더 많은 코어에서 더 빨리 실행하고 소수의 코어에서 최대한의 효율성으로 실행하여 비용을 절감할 수 있습니다. 시간 민감도와 비용 민감도가 균형을 이루는 위치를 찾아 워크로드를 실행하십시오.

정의

클라우드의 비용 최적화에는 4 가지 모범 사례 영역이 있습니다.

- 비용 효율적인 리소스
- 수요와 공급 일치
- 지출 인식
- 시간별 최적화

수요와 공급 일치, **지출 인식** 및 **시간별 최적화** 영역은 AWS Well-Architected 프레임워크 백서에 설명되어 있습니다.

Archived

모범 사례

비용 효율적인 리소스

> HPCCOST 1: 비용 최적화를 위해 워크로드에 사용 가능한 컴퓨팅 및 스토리지 옵션을 어떻게 평가했습니까?
>
> HPCCOST 2: 작업 완료 시간과 비용 간의 균형을 어떻게 평가했습니까?

시스템에 적절한 인스턴스, 리소스 및 기능을 사용하는 것은 비용 절감의 핵심 요소입니다. 어떤 인스턴스를 선택하는지에 따라 HPC 워크로드의 전체 실행 비용이 늘어나거나 줄어들 수 있습니다. 예를 들어 강결합 HPC 워크로드를 여러 대의 작은 서버에서 실행하는 경우 실행하는 데 5 시간이 걸릴 수 있습니다. 반면, 소수의 대용량 서버 클러스터에서 실행하는 경우 실행 비용은 2 배가 되지만 결과를 1 시간 안에 계산할 수 있으므로 전체적으로 비용이 절감됩니다. 스토리지 선택 또한 비용에 영향을 미칠 수 있습니다. 작업 처리 시간 최적화와 비용 최적화를 절충해야 할 수 있음을 감안하고 다양한 인스턴스 크기 및 스토리지 옵션으로 워크로드를 테스트하여 비용을 최적화하는 것이 좋습니다.

AWS 는 유연하고 비용 효율적인 요금 옵션을 매우 다양하게 제공하므로 요구 사항에 가장 적합한 방식으로 EC2 및 다른 서비스의 인스턴스를 사용할 수 있습니다. 온디맨드 인스턴스의 경우 최소 약정이 없으며 시간 단위로 컴퓨팅 파워 비용을 지불합니다. 예약 인스턴스의 경우에는 온디맨드 요금보다 저렴한 요금으로 용량을 예약할 수 있습니다. 스팟 인스턴스의 경우 미사용 Amazon EC2 용량을 온디맨드 요금보다 할인된 요금으로 제공합니다.

Well-Architected 시스템에는 가장 비용 효율적인 리소스가 사용됩니다. 전처리 및 후처리에 관리형 서비스를 사용하여 비용을 절감할 수도 있습니다. 예를 들어 완료된 실행 데이터의 저장 및 후처리를 위한 서버를 유지하는 대신 데이터를 Amazon S3 에 저장한 후 Amazon EMR 또는 AWS Batch 를 사용하여 후처리할 수 있습니다.

많은 AWS 서비스가 비용 절감에 도움이 되는 기능을 제공합니다. 예를 들어 Auto Scaling 을 EC2 와 통합하면 워크로드 수요에 따라 인스턴스를 자동으로 시작하고 종료할 수 있습니다. FSx for Lustre 는 기본적으로 S3 와 통합되며 S3 버킷의 전체 콘텐츠를 Lustre 파일 시스템으로 제공합니다. 이 통합을 활용하면 최소한의 Lustre 시스템을 프로비저닝하여 즉각적인 워크로드에 사용하고 장기 데이터를 비용 효율적인 S3 스토리지에 유지하여 스토리지 비용을 최적화할 수 있습니다. S3 는 다양한 스토리지 클래스를 제공하므로 가장 비용 효율적인 클래스를 데이터에

Archived

사용할 수 있습니다. Glacier 또는 Glacier Deep 스토리지 클래스를 사용하면 가장 저렴한 비용으로 데이터를 아카이브할 수 있습니다.

다양한 인스턴스 유형, 스토리지 요구 사항 및 아키텍처를 사용하여 최적화를 실험하면 최소한의 비용으로 원하는 성능을 유지할 수 있습니다.

수요와 공급 일치

수요와 공급 일치 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

지출 인식

지출 인식 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

시간별 최적화

시간별 최적화 모범 사례 영역에서 HPC 에 고유한 모범 사례는 없습니다. AWS Well-Architected 프레임워크 백서에서 해당하는 섹션을 검토하십시오.

결론

이 렌즈는 클라우드의 고성능 컴퓨팅 워크로드에 사용할 안정적이고 안전하며 효율적이고 경제적인 시스템의 설계 및 운영에 대한 아키텍처 모범 사례를 제공합니다. 이 백서에서는 모범이 되는 HPC 아키텍처와 중요한 HPC 설계 원칙에 대해 살펴봤습니다. 기존 또는 제안된 HPC 아키텍처를 검토하는 데 도움이 되는 일련의 질문을 제공하면서 HPC 렌즈를 통해 5 가지 Well-Architected 기반을 다시 설명했습니다. 이 프레임워크를 아키텍처에 적용하면 안정적이고 효율적인 시스템을 구축하여 HPC 애플리케이션을 실행하고 전문 영역의 범위를 넓히는 데 집중할 수 있습니다.

Archived

기여자

다음은 본 문서 작성에 도움을 준 개인 및 조직입니다.

- Aaron Bucher: HPC 수석 전문 솔루션스 아키텍트, Amazon Web Services
- Omar Shorbaji: 글로벌 솔루션스 아키텍트, Amazon Web Services
- Linda Hedges: HPC 애플리케이션 엔지니어, Amazon Web Services
- Nina Vogl: HPC 전문 솔루션스 아키텍트, Amazon Web Services
- Sean Smith: HPC 소프트웨어 개발 엔지니어, Amazon Web Services
- Kevin Jorissen: 솔루션스 아키텍트 – 기후 및 날씨, Amazon Web Services
- Philip Fitzsimons: Well-Architected 부문 선임 관리자, Amazon Web Services

추가 자료

추가 정보는 다음을 참조하십시오.

- AWS Well-Architected 프레임워크[12]
- https://aws.amazon.com/hpc
- https://d1.awsstatic.com/whitepapers/Intro_to_HPC_on_AWS.pdf
- https://d1.awsstatic.com/whitepapers/optimizing-electronic-design-automation-eda-workflows-on-aws.pdf
- https://aws.amazon.com/blogs/compute/real-world-aws-scalability/

문서 개정

날짜	설명
2019 년 12 월	부 업데이트
2018 년 11 월	부 업데이트
2017 년 11 월	초판 발행

Archived

Notes

[1] https://aws.amazon.com/well-architected

[2] https://d0.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

[3] https://aws.amazon.com/batch/

[4] https://aws.amazon.com/ec2/

[5] https://aws.amazon.com/ec2/spot/

[6] https://aws.amazon.com/message-queue

[7] https://aws.amazon.com/ec2/instance-types/#instance-type-matrix

[8] https://aws.amazon.com/intel/

[9] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html

[10] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSOptimized.html#ebs-optimization-support

[11] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html

[12] https://aws.amazon.com/well-architected

[13] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa.html

Archived