

# 석사학위논문

## 안정성과 비용효율성을 고려한 다중 스팟 인스턴스 추천 시스템 제안

Multiple Spot Instances Recommendation System Towards Stable  
and Cost-Effective Allocation

국민대학교 일반대학원

컴퓨터공학과 컴퓨터공학전공

김 경 환

2024

# 차례

<b>그림 차례</b>	iii
<b>표 차례</b>	iv
<b>국문 요약</b>	v
<b>제 1 장 서론</b>	1
<b>제 2 장 스팟 인스턴스와 가용성</b>	3
2.1 기존 가용성 점수 정보의 한계 . . . . .	4
<b>제 3 장 다중 노드 스팟 인스턴스의 안정성 데이터 수집</b>	6
3.1 Uniform Spacing Query Sampling . . . . .	6
3.2 Tracing Score Transition Point . . . . .	7
3.2.1 쿼리 부담을 줄이기 위한 Caching 와 Early Stop . . . . .	7
<b>제 4 장 다중 노드 스팟 인스턴스 추천</b>	9
4.1 스팟 인스턴스 비용 점수화 . . . . .	9
4.2 스팟 인스턴스 가용성 정량화 . . . . .	10
4.3 최종 점수 계산과 추천 방법 . . . . .	11
<b>제 5 장 Sandevistan 서비스 구현</b>	13
<b>제 6 장 다중 노드 스팟 인스턴스 데이터셋 분석</b>	15
<b>제 7 장 성능 평가</b>	18
7.1 다중 노드에 걸친 SPS 쿼리 방법의 효율성 . . . . .	18
7.2 가용성 점수의 평가 . . . . .	21
7.3 인스턴스 추천 시스템의 성능 . . . . .	23
<b>제 8 장 관련논문</b>	26
<b>제 9 장 결론</b>	28
<b>감사의 글</b>	29

<b>참고 문헌</b>	<b>30</b>
<b>영문 요약</b>	<b>42</b>
<b>부록</b>	<b>43</b>

## 그림 차례

그림 1 단일 노드의 SPS 점수가 높은 경우, 서로 다른 노드 수를 요 청할 때 요청한 노드 수를 만족한 요청의 비율 . . . . .	4
그림 2 T3 변화에 따른 시간별 면적, 기울기, 표준 편차를 사용하여 스팟 인스턴스 가용성 점수를 정량화하는 예시 . . . . .	11
그림 3 Sandevistan 서비스의 전체 구조도 . . . . .	13
그림 4 다양한 인스턴스 카테고리에서 서로 다른 수의 타겟 노드에 따른 SPS 점수 분포 . . . . .	15
그림 5 인스턴스당 코어 수가 다른 노드들로 구성된 컴퓨팅 노드 풀 을 구성할 때, 총 타겟 CPU 코어 수에 따른 SPS의 차이 . . . . .	16
그림 6 같은 인스턴스 유형의 가용 영역에 따른 최대, 최소 T3의 차 이의 분포 . . . . .	17
그림 7 SPS 쿼리 오버헤드를 줄이기 위한 USQS의 효과 . . . . .	19
그림 8 제안된 쿼리 오버헤드 최소화 방법론의 효과 . . . . .	20
그림 9 가용성 점수에 따른 생존율 분포 비교 . . . . .	22
그림 10 Sandevistan과 AWS SpotFleet 및 단일시점 을 이용했을 때 가 용성과 비용 효율성 비교 . . . . .	23
그림 11 T3와 T2로 계산한 안정성 점수의 상관 계수의 절대값 분포. .	45

## 표 차례

테이블 1 쿼리 표기법 . . . . .	43
------------------------	----

## 국문 요약

클라우드 제공업체들은 잉여 자원 활용을 극대화하기 위해 최대 90% 할인된 가격으로 스팟 인스턴스를 제공한다. 그러나 스팟 인스턴스는 갑작스러운 중단 위험이 있다. 전통적인 가격 데이터셋이 이러한 중단 위험을 예측하는 데 사용되어 왔으나, 최근의 정책 변화로 인해 그 효과가 약화되었다. 클라우드 제공업체는 스팟 인스턴스 사용을 장려하기 위해 사용자들에게 중단의 위험을 예측할 수 있는 가용성 데이터셋을 제공한다. 기존 연구들은 이 데이터를 활용하여 중단 가능성 줄이는 방법을 제안했다. 하지만 기존 연구들은 주로 단일 노드 인스턴스에 초점을 맞추어 최근 클라우드 워크로드에 널리 채택되는 다중 노드 환경의 가용성을 고려하지 못했다. 본 논문에서는 다중 노드 가용성과 가격 데이터를 모두 고려하여 최적의 다중 노드 스팟 인스턴스를 추천하는 Sandevistan 시스템을 제안한다. Sandevistan은 다양한 쿼리 제한을 극복하여 방대한 다중 노드 가용성 데이터를 수집했다. 그리고 철저한 분석을 통해 비용 효율적이고 안정적인 다중 노드 스팟 인스턴스를 추천하는 방법론을 제공한다. 제안된 방법론이 다중 노드 스팟 인스턴스의 가용성과 비용 효율성을 얼마나 잘 반영하는지 평가하기 위해 광범위한 실제 중단 실험을 수행했다. 실험 결과, 추천 방법에 따라 Sandevistan이 기존 다중 노드 스팟 인스턴스를 추천하는 클라우드 서비스인 SpotFleet에 비해 20.8% 더 나은 가용성과 26.3% 더 높은 비용 절감을 달성할 수 있음을 보여주었다.

주제어: 클라우드 컴퓨팅, 스팟 인스턴스, 컴퓨팅 리소스, 다중 노드 인스턴스 최적화, 웹 로그 분석

# 제 1 장 서론

클라우드 제공업체는 인터넷을 통해 컴퓨팅 리소스를 사용자에게 제공한다. 이로 인해 사용자는 물리적 리소스를 직접 관리할 필요가 없어졌으며, 이는 컴퓨팅 리소스 사용 방식을 혁신적으로 변화시켰다. 사용자에게 언제든 컴퓨팅 리소스를 제공하기 위해, 클라우드 제공업체들은 최대 수요를 초과하는 리소스를 유지하며, 이로 인해 사용되지 않는 잉여 리소스가 발생한다. 이러한 미사용 리소스의 활용을 극대화하기 위해, 클라우드 제공업체들은 최대 90% 할인된 가격으로 이를 제공하며, 이를 스팟 인스턴스라고 부른다. 스팟 인스턴스는 AWS, Azure, GCP, Alibaba, IBM을 포함한 대부분의 주요 클라우드 제공업체들이 제공하고 있다.

스팟 인스턴스가 처음 등장하였을 때, 대부분의 클라우드 제공업체들은 컴퓨팅 리소스 수요에 따라 스팟 가격을 동적으로 조정했다. 스팟 가격이 초기 입찰 가격을 초과할 때 스팟 인스턴스가 회수되는데, 이 과정을 인스턴스 중단이라고 한다. 인스턴스 중단은 애플리케이션 신뢰성에 부정적인 영향을 미치며, 이는 사용자들의 스팟 인스턴스 사용을 주저하게 만들었다. 이에 클라우드 제공업체들은 스팟 인스턴스 사용을 독려하고 스팟 인스턴스의 중단 위험을 예측할 수 있도록 하기 위해, 다양한 데이터셋을 제공했다. 가장 주목할 만한 데이터셋 중 하나는 특정 시점의 스팟 가격을 보여주는 스팟 인스턴스 가격 데이터셋이다. 이 데이터셋은 스팟 가격 변화 분석 [1]–[5], 애플리케이션 신뢰성 개선 [6]–[13], 최적의 입찰 가격 추천을 위한 스팟 가격 예측 [14]–[19] 등 다양한 연구에 널리 사용되었다.

스팟 인스턴스 가격 데이터셋은 중단 위험을 예측하는 데 사용되었지만, 2017년 스팟 인스턴스 중단 정책의 변경으로 스팟 가격과 인스턴스 종료 간의 상관관계가 감소했다 [20], [21]. 이에 대응하여, 클라우드 제공업체들은 AWS의 Spot Placement Score(SPS)와 Microsoft Azure의 Spot Placement Recommender와 같은 새로운 가용성 데이터셋을 도입했는데, 이는 스팟 인스턴스의 실시간 가용성을 나타낸다. 다양한 스팟 인스턴스 데이터셋에 쉽게 접근할 수 있도록, SpotLake [22]는 다양한 클라우드 제공업체를 위한 웹 기반 데이터셋 아카이브 서비스를 제공한다. 그러나 스팟 인스턴스 즉시 가용성 데이터셋과 관련하여,

현재는 단일 노드 정보만 제공하고 있어 대규모 다중 노드 환경을 필요로 하는 분산 애플리케이션의 요구를 충족시키지 못하고 있다.

본 논문에서는 많은 노드로 컴퓨팅 리소스 풀을 구축할 때 단일 노드 가용성 점수를 사용하는 것의 한계를 탐구하고 다중 노드에 대한 스팟 인스턴스 데이터셋의 필요성을 제시한다. 이에 다중 노드 가용성 데이터셋을 효율적으로 수집하기 위한 몇 가지 방법론을 제안하며, 이를 공개적으로 이용 가능한 웹 서비스를 개발하였다. 가용성 데이터셋을 사용하여 스팟 인스턴스의 안정성을 정량화하는 가용성 점수를 도입하고, 비용과 가용성을 모두 고려한 스팟 인스턴스 추천 알고리즘을 개발한다. 또한 실제 스팟 인스턴스를 사용한 광범위한 실험을 통해 제안된 시스템의 효율성과 효과성을 입증한다. 특히, 본 논문에서 제안한 알고리즘은 클라우드 제공업체가 내부적으로 이용 가능한 정보를 사용하여 기본적으로 제공하는 AWS Spot Fleet 기능보다 20.8% 더 높은 가용성과 26.3% 더 많은 비용 절감을 제공할 수 있다.

요약하면, 본 논문의 핵심 기여는 다음과 같다.

- 큐리 오버헤드를 완화하면서 다중 노드 스팟 인스턴스 가용성 데이터셋을 수집하기 위한 새로운 방법론을 제안했다.
- 스팟 인스턴스 신뢰성을 모델링하는 데 다중 노드 가용성 데이터셋 사용의 효과성을 밝혔다.
- 가용성과 비용을 반영한 다중 노드 스팟 인스턴스 추천 시스템을 제안하고, 실제 광범위한 실험을 통해 그 유효성을 입증했다.
- 수집된 다중 노드 가용성 데이터셋을 제공하는 공개적으로 이용 가능한 웹 서비스를 개발했다.

## 제 2 장 스팟 인스턴스와 가용성

클라우드 제공업체들은 사용자에게 유연한 컴퓨팅 옵션을 제공하기 위해 최대 수요를 초과하는 잉여 리소스를 유지한다. 이러한 잉여 리소스는 스팟 인스턴스로 판매되며, 온디맨드 인스턴스 대비 최대 90% 할인된 가격으로 제공되어 상당한 비용 절감 효과를 제공한다. 그러나 스팟 인스턴스의 가용성은 전반적인 컴퓨팅 리소스 사용량에 따라 변동될 수 있다. 수요가 급증하여 컴퓨팅 리소스가 부족해질 경우, 클라우드 제공업체는 이를 회수할 수 있다. 이러한 회수는 서비스 중단으로 이어질 수 있어, 중요한 애플리케이션 운영에 잠재적 위험 요소가 될 수 있다.

스팟 인스턴스의 동적으로 변화하는 가격과 예상할 수 없는 중단 위험으로 인해, 스팟 인스턴스의 가격과 가용성 모두 사용자에게 중요한 고려사항이다. 클라우드 제공업체들은 스팟 인스턴스의 사용을 독려하기 위해 스팟 가격과 가용성에 대한 데이터셋을 제공한다. 이러한 데이터셋은 특히 입찰 기반 모델 (예: AWS [23], Azure [24], Alibaba [25])에서 중단 위험을 추정하는 데 사용되었다 [14], [15]. 예를 들어, AWS는 정기적으로 스팟 가격을 업데이트하고 과거 가격 데이터를 제공하여 스팟 가격에 기반한 안정적인 스팟 인스턴스 사용에 대한 연구가 이루어졌다 [1], [3], [16]. 그러나 최근의 정책 변경으로 스팟 가격과 중단 간의 상관관계가 감소했고 스팟 가격 업데이트 빈도가 줄어들어 이전 연구의 효과성이 감소하였다 [20], [21]. 이에 대응하여 클라우드 제공업체들은 이제 스팟 인스턴스 가용성과 관련된 데이터셋을 제공한다. AWS와 Azure는 지난 한 달간의 중단 비율과 AWS SPS [26], Azure의 Spot Placement Recommender [27]와 같은 실시간 가용성 데이터를 제공한다. 이 지표는 내부적으로 어떠한 방식으로 계산되는지 공개되지 않았지만, 즉각적인 스팟 인스턴스 가용성을 나타내는 지표로 예를 들어, AWS의 SPS는 각 스팟 인스턴스 유형에 1, 2, 3의 3가지 점수를 할당하며, 높은 점수는 더 큰 가용성을 나타낸다. 사용자는 관리 콘솔이나 API를 통해 이러한 데이터셋에 접근할 수 있지만, 이러한 데이터를 쿼리하는데 상당한 제한이 있다. 이러한 제약을 극복하고 데이터에 더 쉽게 접근할 수 있도록 SpotLake라는 웹 아카이빙 서비스가 개발되었다 [22].

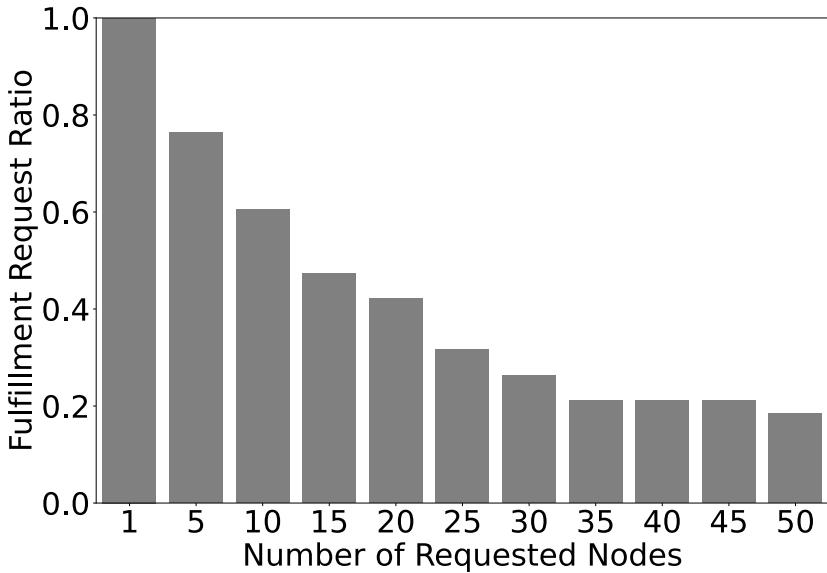


그림 1: 단일 노드의 SPS 점수가 높은 경우, 서로 다른 노드 수를 요청할 때 요청한 노드 수를 만족한 요청의 비율

## 2.1 기존 가용성 점수 정보의 한계

현재 SpotLake는 단일 노드 스팟 인스턴스에 대한 SPS 정보만 제공하며 여러 인스턴스를 동시에 요청하는 시나리오에 대한 SPS 점수, 즉 다중 노드 스팟 인스턴스에 대한 SPS 정보는 제공하지 않는다. 그러나 현대의 클라우드 애플리케이션은 대규모 딥 러닝 모델 분산 훈련과 같은 여러 노드를 가진 분산 환경을 필요로 한다 [28]. 마찬가지로 빅데이터 처리 [29]도 대규모 컴퓨팅 리소스에 의존하며, 이는 상당한 비용을 초래할 수 있다.

이러한 비용을 줄이기 위해, 분산 시스템에 스팟 인스턴스를 활용하는 방법에 대한 관심이 증가하고 있다. 분산 환경에서의 DNN 훈련 [30]–[34], MapReduce를 사용한 빅데이터 처리 [35], 일반 웹 서비스 [9], [11], [13] 등 다양한 분야에 대한 연구가 활발히 진행되고 있다. 그러나 대부분의 연구는 중단 처리에 초점을 맞추고 있으며, 사전에 중단 위험이 적은 스팟 인스턴스를 선택하는 방법을 다루는 방법으로 김경환 외 1인의 연구는 AWS의 데이터셋에서 SPS 값을 예측하여 스팟 인스턴스의 가용성을 향상시키는 방법을 제안했지만 [36], 단일 노드 SPS만 다루고 있어 분산 환경에서의 적용

가능성이 제한적이다.

현재 SpotLake에서 제공하는 단일 노드 SPS가 다중 노드 스팟 인스턴스를 요청할 때 가용성을 나타낼 수 있는지 탐구하기 위해, 다중 노드에 대한 SPS 점수를 고려하지 않고 단일 노드 SPS 점수가 3(가장 높음)인 32개의 인스턴스 유형을 무작위로 선택하고, 다양한 인스턴스 요청에 대한 성공률을 관찰했다. 그림 1에서 가로축은 요청된 인스턴스 수를 나타내고, 세로축은 성공적으로 할당된 요청의 비율을 보여준다. 단일 인스턴스 요청의 경우 성공률이 100%로 단일 노드 SPS 값은 잘 반영하지만, 인스턴스 수가 증가함에 따라 크게 감소한다. 50개 인스턴스의 경우 단 20%의 요청만 성공적으로 할당되었다. 이는 단일 노드 SPS가 여러 스팟 인스턴스를 요청할 때 가용성의 신뢰할 수 있는 지표가 아님을 보여준다.

## 제 3 장 다중 노드 스팟 인스턴스의 안정성 데이터 수집

스팟 인스턴스를 사용하여 비용 효율적이고 신뢰할 수 있는 대규모 리소스 풀을 구축하기 위해서는 적절한 인스턴스 유형과 인스턴스를 할당할 지역을 선택하는 것이 필수적이다. SpotLake는 단일 노드에 대한 SPS 값을 제공하지만, 이는 다중 노드 환경에는 충분하지 않다. AWS는 여러 인스턴스에 대한 SPS 값을 제공하지만, 쿼리 제한으로 인해 이 데이터를 효율적으로 쿼리하기 어렵다. 예를 들어, 24시간 내에 오직 50개의 고유한 쿼리를 구성만 허용되며, 동일한 구성에 대해 노드 수가 다른 쿼리는 별도의 요청으로 취급된다. 이러한 제한으로 인해 모든 인스턴스 유형, 지역, 노드 수 조합에 대한 데이터를 수집할 때 상당한 쿼리 오버헤드가 발생한다. SpotLake가 제안한 최적의 쿼리 방식 [22]을 사용하더라도, 단일 노드에 대한 모든 인스턴스 유형의 경우 66개의 별도 계정을 통해 약 3,300개의 쿼리가 필요하다. 게다가 다중 노드를 쿼리하려면 쿼리 오버헤드와 필요한 계정 수가 비례적으로 증가한다. 예를 들어, 1개에서 50개 노드까지의 SPS 값을 쿼리하려면 3,330개의 계정에 걸쳐 165,000개의 쿼리가 필요하다. 더욱이 변화 하는 데이터셋을 탐지하려면 이러한 쿼리를 주기적으로 실행해야 하므로, 모든 가능한 조합을 쿼리하는 것은 불가능에 가깝다. 이를 해결하기 위해 본 논문은 데이터셋을 효율적으로 쿼리하기 위한 두 가지 방법론을 제안한다.

### 3.1 Uniform Spacing Query Sampling

다중 노드 스팟 인스턴스에 대한 SPS 값을 쿼리할 때 발생하는 오버헤드를 완화하기 위해, *Uniform Spacing Query Sampling* (USQS) 방법을 제안한다. 매 쿼리 주기마다 각 인스턴스 유형과 지역에 대해 모든 가능한 노드 수를 쿼리하는 대신, USQS는 쿼리 주기 전반에 걸쳐 쿼리할 노드 수를 번갈아 가며 쿼리한다. 구체적으로, SPS 쿼리는 주기적으로  $p$ 분마다 수행된다.  $T_{max}$ 를 최대 노드 수,  $T_{min}$ 을 최소 노드 수,  $T_c$ 를 현재 쿼리되는 노드 수,  $T_s$ 를 쿼리후 증가시킬 노드 수라고 하자.  $T_c$  노드에 대한 쿼리 후,  $p$ 분 후에 발생하는 다음 쿼리는  $T_c + T_s$  개의 노드를 대상으로 할 것이다. 쿼리 노드 수가  $T_{max}$ 를 초과하면 쿼리는  $T_{min}$ 으로 재설정되고, 이 과정은 각 주기마다  $T_s$ 씩 증가하며 계속된다. 이 USQS 방법은 쿼리 간격( $p$ )을 조정함으로써 쿼리 오버헤드를 유연하게 제어할 수 있게

한다. 그러나 주기당 하나의 특정 노드 수만 쿼리되기 때문에, 특정 노드 수에 대한 쿼리는  $(\lfloor \frac{T_{max}-T_{min}}{T_s} \rfloor + 1) \times p$  분 후에 반복되며, 이 기간 동안 변화된 SPS 값을 수집하지 못한다.

## 3.2 Tracing Score Transition Point

SPS 점수는 동일한 가용 영역 내에서 주어진 인스턴스 유형에 대해 노드 수가 증가함에 따라 일정하거나 감소한다. 이러한 관찰을 바탕으로, 노드 수가 증가함에 따라 SPS 점수의 변화를 감지하는 방법을 소개하며, 이를 Tracing Score Transition Point(TSTP)이라고 부른다.  $T_{min}$ 과  $T_{max}$  사이의 노드 수에 대해, SPS 점수가 3에서 2 또는 1로 전환되는 가장 큰 노드 수를  $T_3$ 로 표시하고, SPS 점수가 2에서 1로 전환되는 가장 큰 노드 수를  $T_2$ 로 표시하며,  $T_3 \leq T_2$ 이다. 이는  $T_3$  이하의 모든 노드 수에 대해 SPS 점수가 항상 3이 되고,  $T_2$ 보다 큰 노드 수에 대해서는 SPS 점수가 1이 되며,  $T_3$ 와  $T_2$  사이의 수에 대해서는 SPS 점수가 2가 됨을 보장한다. 이를 바탕으로 TSTP 알고리즘은 모든 노드 수에 대해 SPS 점수를 쿼리하지 않고도  $T_3$ 와  $T_2$ 를 효율적으로 식별하기 위해 표준 이진 탐색 알고리즘을 적용한다. 이진 탐색을 사용하면 최대  $O(\log(T_{max} - T_{min}))$  쿼리로 SPS 값 변화 지점을 감지할 수 있다.

### 3.2.1 쿼리 부담을 줄이기 위한 Caching 와 Early Stop

SPS가 일반적으로 급격히 변동하지 않는다는 점을 고려하여 [22], 표준 이진 탐색의 쿼리 오버헤드를 줄이기 위해 이전 쿼리 주기의 결과를 활용하는 방법을 제안한다. 이전 주기에서 감지된  $T_3$ 와  $T_2$  값을 캐시하고, 다음 쿼리 반복에서는 먼저 이전에 기록된 SPS 근처 범위 내에서 변화가 있는지 확인한다. 캐시된  $T_3$  값을 활용하면 단일 쿼리로 쿼리 범위를 크게 줄일 수 있다. 그러나 정확한 변곡점을 식별하려면 후반 단계에서 추가 쿼리가 필요하다는 한계가 여전히 존재한다. 이진 탐색의 초기 단계에서는 탐색 범위가 크게 감소하지만, 후반 단계에서는 감소가 점진적으로 이루어진다. 제안한 접근 방식에서도 이와 유사하게 초기에는  $T_3$ 와  $T_2$  값으로 빠르게 수렴하지만, 후반 단계에서는 이미 실제  $T_3$ 와  $T_2$ 에 근접해 있더라도 정확한 값을 찾기 위해 추가적인 쿼리가 계속 진행된다.

다.  $T_3$ 와  $T_2$ 의 정확한 값을 알 수 없지만, 허용 가능한 오차 범위 내의 근사값을 갖는 것만으로도 여러 스팟 인스턴스의 안정성을 평가하는 데 큰 도움이 될 수 있다. 이를 바탕으로, 정확한 변경 지점을 차운 대신 쿼리 횟수를 줄이는 조기 종료 메커니즘을 제안한다. 쿼리 범위의 너비가 사전 정의된 임계값  $e$  이하일 때 이진 탐색을 종료하고, 이 시점에서 노드 수를 현재 쿼리 범위의 하한으로 설정한다.

이 알고리즘은 알고리즘 1에 설명되어 있으며, 표기법은 부록의 표 1에 요약되어 있다.

## 제 4 장 다중 노드 스팟 인스턴스 추천

최적의 스팟 인스턴스를 선택할 때는 가용성과 비용을 모두 고려해야 한다. 이는 다목적 최적화 문제를 사용하여 해결할 수 있다 [37]. 다목적 최적화 문제는 다양한 여러 목적 함수들을 사용자 정의 매개변수를 기반으로 함수들의 중요도를 조정하고 하나로 결합하여, 최적의 결과를 얻는 것을 말한다. 이 장에서는 가용성 점수와 비용 점수를 가중치 매개변수와 함께 정량화하여 다중 노드 스팟 인스턴스를 추천하는 알고리즘을 소개한다.

### 4.1 스팟 인스턴스 비용 점수화

스팟 인스턴스의 비용 비교를 위한 정량적 지표를 수립했다.  $C$ 를 모든 대상 스팟 인스턴스의 가격 데이터 집합으로,  $c_i$ 를 개별 인스턴스의 가격 데이터로 표시한다. 먼저, 전체 가격 데이터  $C$  내에서 각 스팟 인스턴스  $c_i$ 의 가격에 대해 MinMax 정규화 [38]를 수행했다. 정규화된 가격  $nc_i$ 를 사용하여 비용 점수,  $CS_i$ 를 다음과 같이 계산한다:

$$CS_i = 100 \times \left(1.0 - \frac{1}{1 + e^{-12(nc_i - 0.5)}}\right) \quad (1)$$

스팟 인스턴스 가격의 경우, 일부 인스턴스가 다른 스팟 인스턴스에 비해 현저히 저렴하거나 비쌀 수 있다. 이러한 극단적 값들을 기반으로 가격 점수를 선형적으로 계산하면, 전체 시스템 내에서 비용 점수의 가중치가 굉장히 커질 수 있다. 이를 방지하고 가용성과 비용 절감을 모두 고려하는 시스템을 만들기 위해, 극단적 값들의 영향을 균형 잡힌 비용 점수로 완화하는 시그모이드 함수를 적용했다 [39]. 시그모이드 함수의 하이퍼파라미터를 경험적으로 설정하여, 저렴한 가격에는 100에 가까운 점수를, 비싼 가격에는 0에 가까운 점수를 할당했다.

## 4.2 스팟 인스턴스 가용성 정량화

스팟 인스턴스의 가용성을 정량적으로 평가하기 위해, SPS 점수가 3으로 유지되는 최대 노드 수( $T3$ )에 초점을 맞춰 가용성 점수를 계산했다. 다양한 시나리오에서  $T3$ 와  $T2$  값의 시간적 특성을 분석한 결과, 부록의 그림 11에서 보이듯이 두 값 사이에 종속적 관계가 있음을 확인했다. 이는  $T3$  노드 수만 사용해도 충분함을 시사한다.  $T3$ 의 시간적 변화를 고려하기 위해, 이러한 지표의 시계열 그래프를 생성하고 식 2를 사용하여 가용성 점수 계산에 통합했다. 시계열 그래프에서 X축은 시간을, Y축은  $T3$  값을 나타내며, 높은  $T3$ 는 더 큰 가용성을 의미한다.

$$AS_i = 100 \times (A3_i \times (1.0 + 0.1 \times (m_i - \sigma_i))) \quad (2)$$

스팟 인스턴스  $i$ 의 가용성 점수를  $AS_i$ 로 정의했다. 관찰 기간 동안 인스턴스 유형  $i$ 의  $T3$  값을  $T3_i$ 로 표시한다.  $T3_i$ 의 시간적 변화를 기반으로 곡선 아래 면적을 계산하여 이 값을  $A3_i$ 로 표현했고  $T3_i$ 의 적분값을 나타낸다. 계산된 값은 모든 인스턴스에 대해 MinMax 스케일러를 사용하여 [0.0, 1.0] 사이로 정규화했다. 높은 값은 더 큰 가용성을 나타낸다. 시간에 따른  $T3_i$ 의 추세를  $m_i$ 로 표시하고, 변화의 기울기를 사용하여 계산했다. 양의 기울기는 시간이 지남에 따라 가용성이 증가함을 나타내며, 이는 긍정적인 것으로 간주되어  $A3_i$  점수를 최대 10% 증가시킨다. 반대로, 음의 기울기는 가용성이 감소함을 의미하며 최대 10%의 패널티를 부여한다. 기울기  $m_i$ 를 계산하기 위해 1차 선형 회귀 모델 40,  $y = \beta_0 \times x + \beta_1$ 을 사용했다. 여기서  $x$ 는 시간을,  $y$ 는  $T3_i$  값을 나타낸다. 모델의 오차를 최소화하는  $\beta_0$  값을 MinMax 정규화 후 기울기를 결정하는 데 사용했다.  $\sigma_i$ 는 시간에 따른  $T3_i$ 의 변동성과 관련된 패널티를 나타내며, 표준 편차를 사용하여 계산했다. 큰 표준 편차는 스팟 인스턴스  $T3_i$ 의 변동이 더 크다는 것을 의미하며, 이는 가용성에 부정적인 영향을 미친다. 이 표준 편차 값에 따라  $A3_i$  점수에 최대 10%의 패널티가 적용된다.  $T3_i$  값이 변동 없이 안정적으로 유지되면 패널티가 적용되지 않는다. 모든 인스턴스에 대한 표준 편차 값을 계산하고, MinMax 정규화를 적용한 후 점수에 패널티를 반영했다.

그림 2은  $T3$  값의 다양한 패턴에 따라 가용성 점수  $AS$ 가 어떻게 변화하

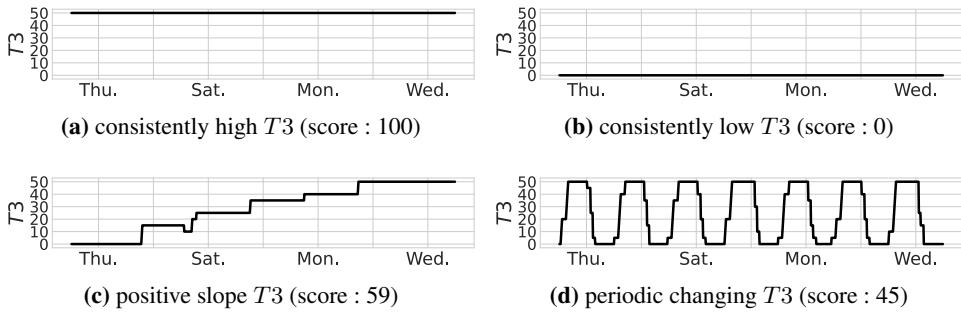


그림 2:  $T3$  변화에 따른 시간별 면적, 기울기, 표준 편차를 사용하여 스팟 인스턴스 가용성 점수를 정량화하는 예시

는지를 보여주는 예시들을 나타낸다. 각 그래프의 가로축은 시간을, 세로축은  $T3$ 를 나타낸다. 이 예시에서 SPS가 수집된 노드의 최대 수는 50으로, Y축의 최댓값이다. 그림 2a는 관찰 기간 동안  $T3$  값이 최대치인 50을 유지하는 시나리오를 보여준다. 결과적으로 적분 면적 기반 점수인 A3는 만점을 받고, 변동이 없으므로 표준 편차에 따른 패널티가 적용되지 않는다. 또한 기울기가 0이므로 보너스 점수도 추가되지 않아 총점은 100점이 된다. 반면, 그림 2b는 기간 내내  $T3$  값이 0을 유지하는 경우를 보여주며, 면적 점수는 0이 되고 최종 점수 역시 0이 된다. 값이 일정하지 않은 경우, 변동 패턴에 따라 보너스와 패널티 점수가 적용된다. 예를 들어, 그림 2c에서는 변동성으로 인한 표준 편차 패널티가 있지만,  $T3$ 의 상승 추세로 인해 양의 기울기에 대한 보너스 점수가 부여되어 최종 점수에 반영된다. 그림 2d는 규칙적인 변동을 보이는 경우를 나타낸다. 이 경우 표준 편차로 인한 패널티가 부여되고, 기울기가 0에 가까워져 추가 점수는 부여되지 않는다.

### 4.3 최종 점수 계산과 추천 방법

비용 효율적이고 안정적인 스팟 인스턴스를 추천하는 첫 단계는 사용자가 지정한 요구사항을 충족하는 인스턴스 조합을 추출하는 것이다. Sandevistan에서 사용자는 얻고자 하는 총 CPU 코어 수와 최소 총 메모리 크기를 명시적으로 지정해야 한다. 추천 시스템은 먼저 사용자의 조건을 만족하는 모든 인스턴스 유형을 추출한다. 또한 사용자는 선호하는 인스턴스 유형, 카테고리 또는 지역

을 지정할 수 있다. 추천 시스템은 관련된 모든 인스턴스에 대해 가용성 점수 ( $AS_i$ )와 비용 점수( $CS_i$ )를 계산한다. 최종 점수는 사용자가 입력한 매개변수  $W$ 를 사용하여 식 3와 같이 계산된다.  $W$  값은 [0.0, 1.0] 사이로 설정되며,  $W$ 가 증가할수록 가용성 점수  $AS_i$ 에 할당되는 가중치가 따라 증가한다.

$$S_i = W \times AS_i + (1.0 - W) \times CS_i \quad (3)$$

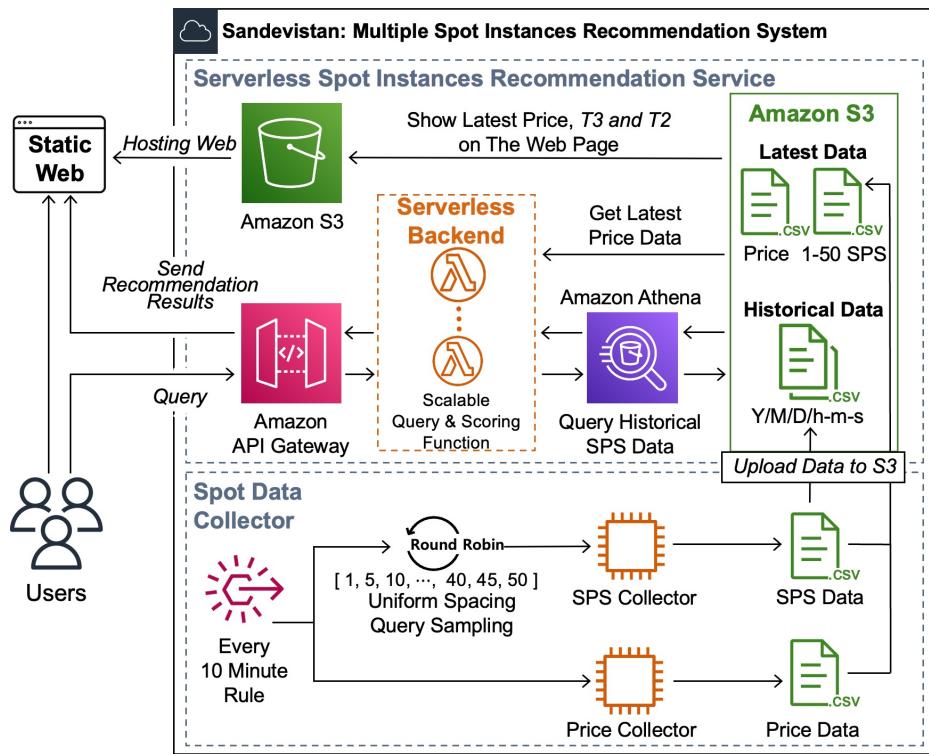


그림 3: Sandevistan 서비스의 전체 구조도

## 제 5 장 Sandevistan 서비스 구현

Sandevistan은 공개적으로 이용 가능한 웹 서비스다. AWS 스팟 인스턴스를 대상으로 구현된 시스템의 구조는 그림 3과 같다. 하단의 *Spot Data Collector*는 최적화된 SPS 쿼리 방법 중 하나인 USQS 기법을 적용한다. 쿼리는 10분마다 주기적으로 수행되며, 대상 노드 수는 1에서 50까지 5씩 증가한다. 가격 데이터의 경우, AWS에서 제공하는 가격 데이터셋을 직접 활용한다. 수집된 데이터셋은 S3에 저장되어 사용자가 직접 접근하거나 추천 모듈에서 사용할 수 있다. 데이터는 연-월-일-시간 이름 규칙으로 저장된다.

추천 모듈은 사용자의 추천 요청을 유연하게 처리하기 위해 서비스 아키텍처 [41]를 사용하여 설계되었다. 사용자가 원하는 최소 CPU 코어 수, 메모리 크기, 안정성 및 비용 효율성 가중치 매개변수와 함께 특정 인스턴스 유형, 지역

과 같은 정보를 제공하면, 추천 서비스의 엔드포인트 역할을 하는 API Gateway 서비스로 전송된다. 이는 Lambda 서비스로 요청을 전달한다. Lambda는 사용자의 요구 사항을 충족하는 인스턴스 유형을 필터링하고 서비스 데이터 쿼리 서비스인 Athena서비스를 통해 대상 인스턴스의 과거  $T_3$  값을 쿼리한다. 이 정보를 사용하여 추천 점수를 계산한 후 사용자에게 반환한다. 웹 서비스는 S3를 통해 정적 HTML 파일로 제공된다.

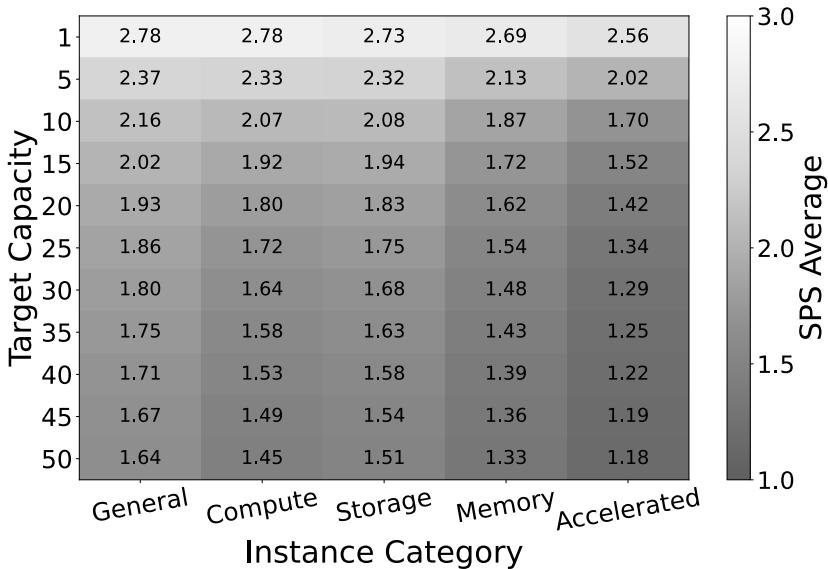


그림 4: 다양한 인스턴스 카테고리에서 서로 다른 수의 타겟 노드에 따른 SPS 점수 분포

## 제 6 장 다중 노드 스팟 인스턴스 데이터셋 분석

이번 장에서는 SPS 데이터 수집 모듈로 수집한 다중 노드 SPS 데이터셋에서 어떤 특성이 관찰되는지 분석하였다.

그림 4는 다양한 인스턴스 카테고리(가로축)와 각 카테고리 내 다양한 대상 노드 수에 대한 평균 SPS 값(세로축)의 변화를 히트맵으로 보여준다. 분석된 데이터는 2024년 8월 12일부터 8월 18일 사이의 데이터를 사용했으며, 총 787 개의 인스턴스 유형을 포함한다. 히트맵에서 밝은 색은 높은 SPS 값을, 어두운 색은 낮은 SPS 값을 나타낸다. 모든 인스턴스 카테고리에서 목표 노드 개수가 증가함에 따라 SPS 값이 감소하는 것이 관찰된다. 이는 여러 스팟 인스턴스를 요청할 때 가용성이 낮아질 수 있다는 예상과 일치한다. 인스턴스 카테고리 중 Accelerated Computing 카테고리의 인스턴스 유형이 가장 낮은 평균 SPS 값을 보이며, 노드 수가 증가함에 따라 급격히 감소한다. 이러한 경향은 딥 러닝의 발전으로 인한 GPU기반 인스턴스 수요 증가로 인한 것으로 보인다 [6]. 다른 인스턴스 카테고리의 경우 SPS 값 감소 정도가 다양하게 나타나, 대규모 자원 풀을 구축할 때 적절한 스팟 인스턴스 유형 선택의 중요성을 보여준다.

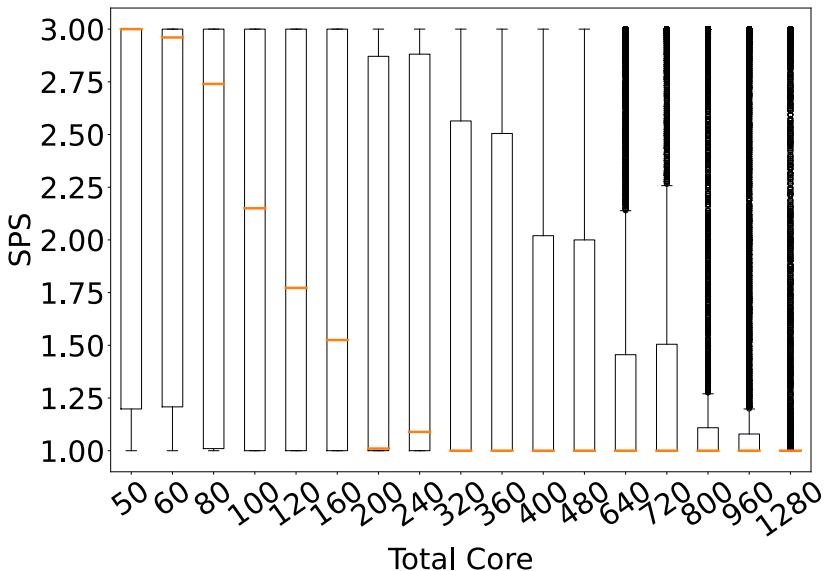


그림 5: 인스턴스당 코어 수가 다른 노드들로 구성된 컴퓨팅 노드 풀을 구성할 때, 총 타겟 CPU 코어 수에 따른 SPS의 차이

여러 인스턴스 노드를 선택할 때, 특정 인스턴스 유형을 선택하는 것보다 대상 클러스터의 원하는 하드웨어 사양(총 CPU 코어, 총 메모리 크기)을 정의한 후 이를 충족하는 안정적인 스팟 인스턴스를 선택하는 것이 더 효과적인 경우가 많다. 이를 보여주기 위해, 특정 유형의 여러 인스턴스를 사용하여 특정 총 CPU 코어 수를 달성할 때의 SPS 값을 분석했다. 그림 5에서 가로축은 목표 총 코어 수를, 세로축은 해당 목표 코어를 달성하기 위한 SPS 값을 나타낸다. 예를 들어, 총 코어 수가 320일 때, 321개의 인스턴스 유형이 최대 50개의 노드를 사용하여 이 총 코어 수를 만족시킬 수 있다(64개 vCPU를 가진 m7i.16xlarge 5개 사용).

SPS 값은 박스 그림 형식으로 표시된다. 총 코어 수가 작을 때는 중앙값 SPS 값이 3.0으로, 비교적 안정적인 스팟 인스턴스를 활용할 수 있음을 나타낸다. 그러나 총 코어 수가 증가함에 따라 SPS 값이 급격히 감소하며, 총 코어 수가 320을 초과하면 중앙값 SPS 값이 최저점인 1.0으로 떨어진다. 그럼에도 불구하고 75% 사분위수, 최대값 또는 이상치 범위에서 여전히 더 높은 SPS 값을 가진 안정적인 인스턴스를 선택할 수 있다. 이러한 결과는 다중 노드 안정성 데이터를 참고하여 스팟 인스턴스 유형 선택하는 것의 중요성을 강조한다.

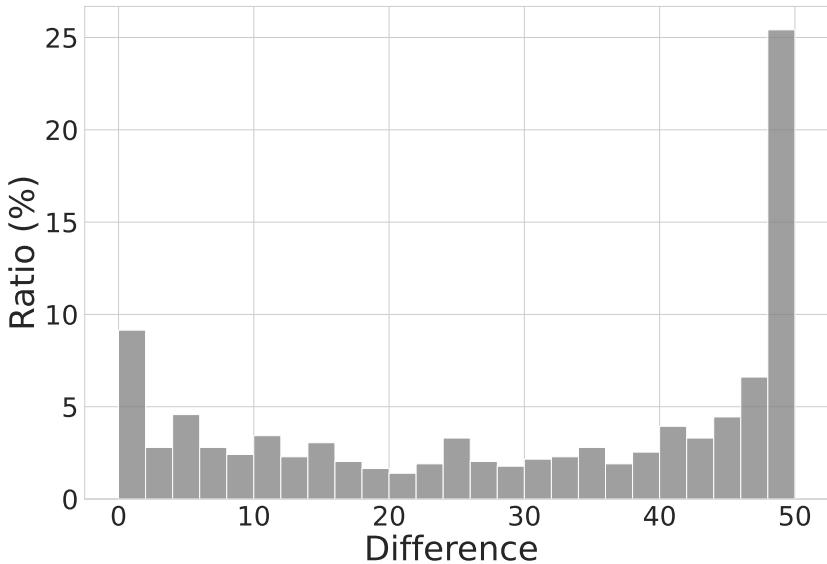


그림 6: 같은 인스턴스 유형의 가용 영역에 따른 최대, 최소 T3의 차이의 분포

다음으로, 지역적 특성에 따른 SPS 값의 분포를 분석한다. 그림 6는 동일한 인스턴스 유형이 다른 지역과 가용 영역에 위치할 때 SPS 값이 3으로 유지되는 최대 노드 수를 나타내는 T3의 차이를 검토한다. 각 인스턴스 유형에 대해 최대 T3와 최소 T3 간의 차이를 계산했다. 큰 차이는 동일한 인스턴스 유형이라도 위치에 따라 안정성에 상당한 차이가 있을 수 있음을 나타낸다. 그림 6에서 가로 축은 그 차이를, 세로축은 비율을 나타낸다. 그림에서 볼 수 있듯이, 25% 이상의 인스턴스 유형에서 그 차이가 관찰된 최대값인 50에 도달하며, 이는 안정적인 스팟 인스턴스를 선택할 때 적절한 지역과 가용 영역을 선택하는 것의 중요성을 강조한다.

## 제 7 장 성능 평가

이번 장에서는 SPS 데이터 수집 모듈의 효율성과 운영 오버헤드를 분석하고, 실제 스팟 인스턴스 환경에서 광범위한 중단 실험을 통해 인스턴스 추천 알고리즘의 효과를 입증한다. 스팟 인스턴스의 가용성과 중단 확률을 평가하는 가장 직관적인 방법은 스팟 인스턴스를 계속 실행하면서 주기적으로 스팟 인스턴스의 상태를 기록하는 것이다 [36], [42], [43]. 이러한 접근 방식은 소규모 실험에는 유효하지만, 실험 규모가 커질수록 비용이 크게 증가할 수 있다. Zhanghao 외 6명이 진행한 연구에서는 스팟 인스턴스의 중단 실험의 비용을 효과적으로 제어하기 위해 스팟 인스턴스를 지속적으로 실행할 필요가 없음을 보여주었다 [44]. 대신, 주기적으로 스팟 인스턴스 요청을 보내고 요청 성공 여부를 관찰함으로써 스팟 인스턴스의 가용성을 효과적으로 모델링할 수 있음을 보여주었다. 본 논문에서는 많은 인스턴스 유형을 포함하는 대규모 실험을 고려하여 주기적으로 스팟 요청을 보내고, 성공 또는 실패를 기록하며, 이러한 관찰을 바탕으로 중단 데이터를 생성했다. 이 결과를 사용하여 다음과 같은 질문에 답하고자 한다.

**질문 1:** 다중 노드에 걸친 SPS의 쿼리 오버헤드를 줄이기 위해 설계된 수집 메커니즘이 효과적인가?

**질문 2:** 제안된 가용성 점수가 스팟 인스턴스의 실제 신뢰성을 정확히 반영하는가?

**질문 3:** 비용과 가용성 데이터를 모두 활용하는 노드 추천 알고리즘이 예상대로 작동하는가?

### 7.1 다중 노드에 걸친 SPS 쿼리 방법의 효율성

우선 질문 1에 답하고자 한다. Sandevistan은 쿼리 오버헤드를 줄이기 위해 *USQS* 데이터 수집 메커니즘을 사용하여 가용성 데이터를 수집했다. 그 효과를 검증하기 위해, 전체 대상 노드 수에 대해 SPS 쿼리를 수행하는 *Full Scan*으로 수집한 데이터와 그 결과를 비교했다. 그림 7은 *Full Scan*과 *USQS* 방법으로 수집한 SPS 값의 차이를 보여준다. 가로축은 쿼리할 대상 노드 수를, 세로축은

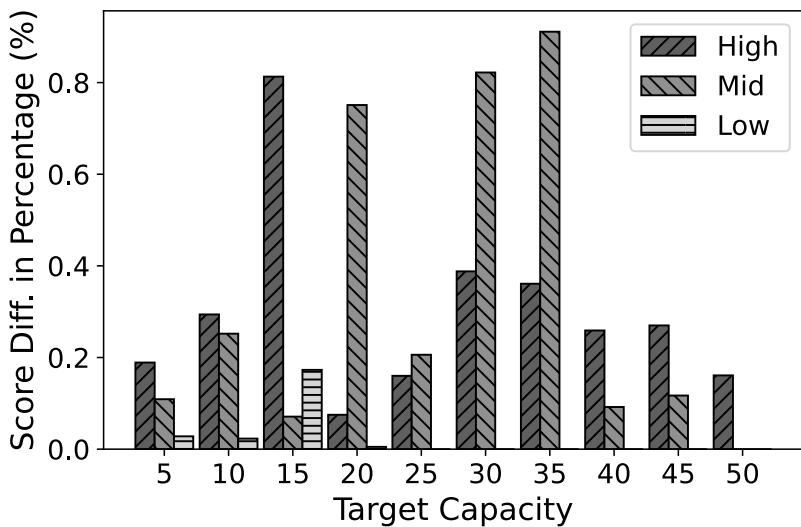


그림 7: SPS 쿼리 오버헤드를 줄이기 위한 USQS의 효과

*Full Scan*과 *USQS*로 수집한 평균 SPS 값의 백분율 차이를 나타낸다. 이 분석에 사용된 데이터는 2024년 8월 7일부터 8월 12일 사이의 데이터를 사용하였으며, 인스턴스 유형과 AZ 쌍은 SPS 값 변동의 표준 편차에 따라 High, Mid, Low 그룹으로 분류되었다. SPS 값 변동의 정도와 관계없이 차이는 미미했으며, 최대 차이는 0.91%였다. 이는 *USQS*를 사용해 100분 간격으로 수집한 데이터와 *Full Scan*을 사용해 10분 간격으로 수집한 데이터를 비교해도 SPS 값의 차이가 미미하며, *USQS*로 수집한 데이터를 추가 분석에 사용해도 타당함을 나타낸다.

다음으로 그림 8에서 제안된 SPS 쿼리 오버헤드 감소 방법의 효과를 검증한다. 세로축은 *Full Scan*으로 얻은 실제  $T_3$  값과 각기 다른 쿼리 방법으로 얻은  $T_3$ 값의 차이를 박스 그림으로 나타낸다. 낮은 값일수록 실제  $T_3$ 값을 더 잘 나타냄을 의미한다. 오른쪽 세로축은 수집 주기당 인스턴스 유형별 필요한 평균 쿼리 수를 나타내며, 이는 별 기호로 표시된다. *Uniform Spacing*은 USQS를 가리키며, 주기당 단일 노드 수의 안정성 데이터만 쿼리하여 단 하나의 쿼리만 필요로 한다. 그러나 각 주기마다 다른 노드 수의 안정성 데이터 쿼리하기 때문에, SPS 값이 빠르게 변동하는 인스턴스 유형의 경우 실제  $T_3$ 값과 차이가 있을 수 있다. 중앙값 차이는 2이지만, 100분 쿼리 간격으로 인해 드물게 큰 편

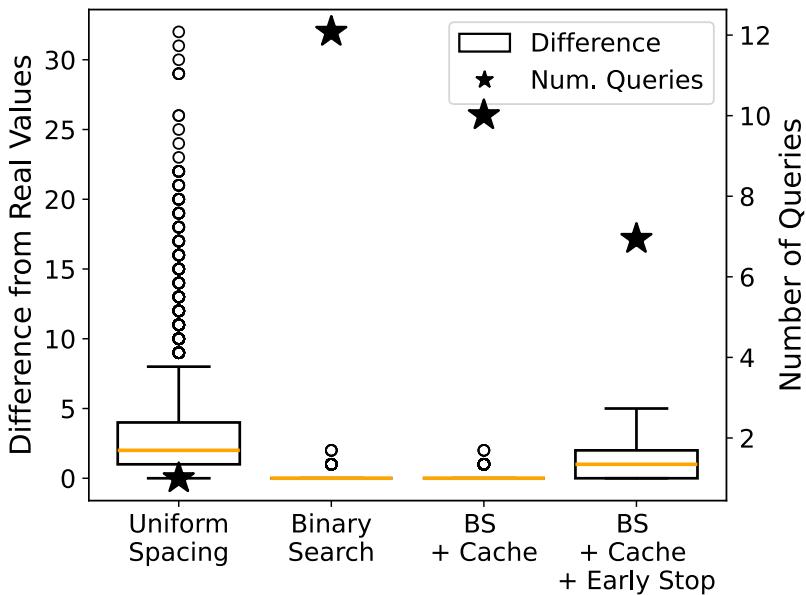


그림 8: 제안된 쿼리 오버헤드 최소화 방법론의 효과

차가 발생할 수 있으며, 이는 32로 나타났다. 주로 이진 탐색을 사용하는 TSTP 방법의 경우, 최적화 기법을 적용하지 않았을 때 10분 간격당 평균 쿼리 수는 12로 관찰되었다. 하지만 이 방법은 각 주기마다  $T_3$  값을 정확히 식별하기 때문에 Full Scan과의 차이가 미미했다. 차이가 없어야 하지만 약간의 편차가 관찰되었는데, 이는 측정 시간의 미세한 차이 때문일 것이다. *Binary Search + Cache* 방법을 도입하여 캐시된  $T_3$  값을 이용해 이진 탐색 범위를 좁힘으로써 평균 쿼리 수를 10.01로 줄였다. 이를 통해  $T_3$  값을 정확히 식별하면서도 더 적은 쿼리로 가능했다. 마지막으로, *Caching*과 *Early Stop*을 적용할 때, 정지 범위  $e$ 는 광범위한 실험을 바탕으로 경험적으로 4로 설정되었다. 이 방법은 평균 쿼리 수를 6.94로 줄여 이진 탐색 기반 알고리즘 중 가장 낮은 쿼리 오버헤드를 보였다. 조기 종료로 인해 다른 이진 탐색 방법에 비해  $T_3$ 에 약간의 차이가 있을 수 있지만, 평균 차이는 0.905에 불과해 큰 의미는 없다. 이 실험은 여러 스팟 인스턴스에서 SPS 값을 추적하는 데 있어 USQS와 TSTP 방법의 효율성을 입증했다. USQS는 쿼리 오버헤드를 크게 줄이고, TSTP는 정확한  $T_3$  추적을 보장한다. 두 방법 모두 정확성과 효율성을 결합하여 실제 SPS 데이터 수집에 균형 잡힌 접근 방식을 제공한다.

## 7.2 가용성 점수의 평가

다음으로 질문 2에 답하고자 한다. 다양한 시나리오에서 여러 스팟 인스턴스의 가용성을 평가하기 위해, 이전 3일간의  $T_3$  값을 사용하여 계산된 가용성 점수를 균등하게 분포시켜 다른 지역에 위치한 100개의 인스턴스 유형을 선택했다. 실험은 2024년 9월 13일부터 10월 9일 사이 24시간 동안 진행되었으며, 인스턴스 유형별로 50개의 스팟 인스턴스를 동시에 요청했다.

가용성 실험 데이터를 수집한 후, 스팟 인스턴스의 생존 시간을 정량적으로 비교하기 위해 Kaplan-Meier Estimator [45](KME)와 Cox proportional hazard model [46]을 사용했다. 이 두 가지 통계적 방법은 수명 데이터를 기반으로 생존율을 추정하거나 특정 변수가 생존에 미치는 영향을 분석하는데 사용된다 [47]–[49]. 주로 의학 분야에서 약물 투여 후 환자의 생존을 평가하거나 비즈니스에서 고객 이탈 요인을 분석하는 데 사용된다. 이러한 관련성을 고려할 때, 이 모델들이 스팟 인스턴스의 생존 시간을 분석하는 데 적합하다고 판단했다. Cox proportional hazard model의 위험비는 다음과 같이 계산된다.

$$h(t|x) = h_0(t) \exp((x - \bar{x})'\beta)$$

$h(t|x)$ 는 가용성 점수  $x$ 가 주어졌을 때 시간  $t$ 에서의 위험비를 나타내고,  $h_0(t)$ 는 가용성 점수를 고려하지 않은 스팟 인스턴스 중단 확률을 반영하는 기준 위험비이다.  $x$ 는 개별 가용성 점수,  $\bar{x}$ 는 모든 유형의 평균 가용성 점수,  $\beta$ 는 가용성 점수가 스팟 중단 가능성에 미치는 영향을 정량화하는 회귀 계수이다. 최적의  $\beta$ 는 로그 우도 함수 [50]를 사용하여 스팟 중단에 가장 잘 맞도록 추정된다.

분석 결과, 가용성 점수는 전체 생존비와 유의한 상관관계가 있음을 나타냈다( $P \leq 0.05$ ). 가용성 점수에 대한 위험비는 0.9903(95% 신뢰구간: 0.9899 - 0.9907)이었다. 이는 가용성 점수가 1점 증가할 때마다 중단 위험이 약 0.97% 감소함을 의미하며,  $e^{-0.0097 \times \Delta x}$  패턴을 따른다. 구체적으로, 가용성 점수가 100 일 때 중단 위험은 가용성 점수가 0일 때에 비해 약 62.1% 감소한다. 이는 가

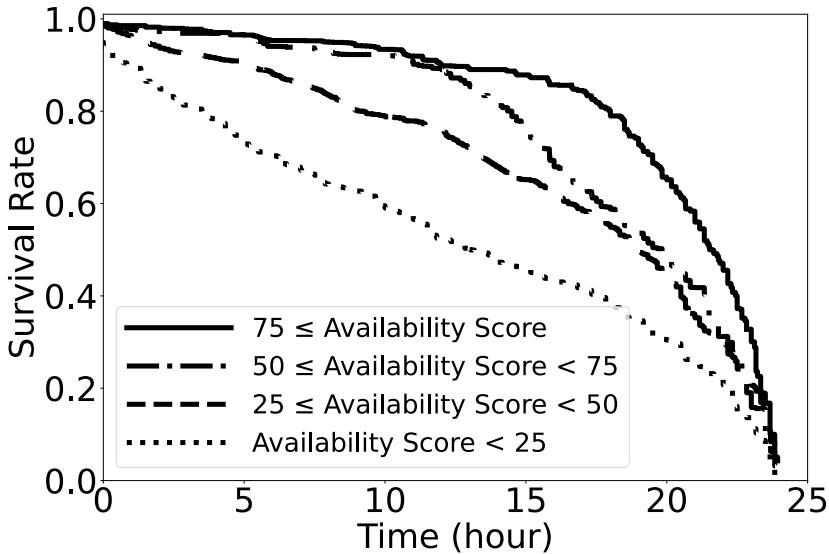


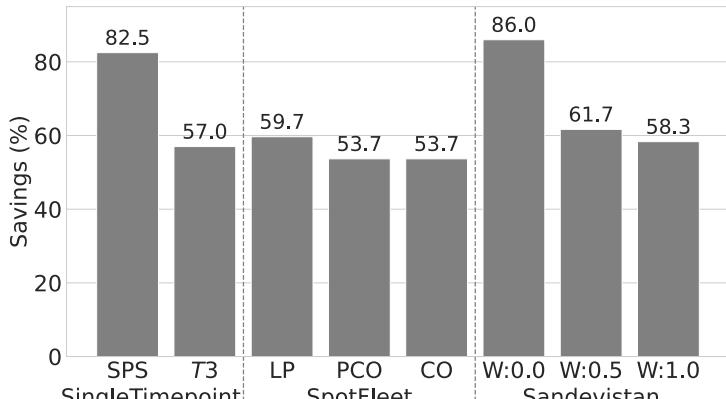
그림 9: 가용성 점수에 따른 생존율 분포 비교

용성 점수가 스팟 인스턴스의 가용성을 모델링하는 데 있어 중요한 지표임을 나타낸다.

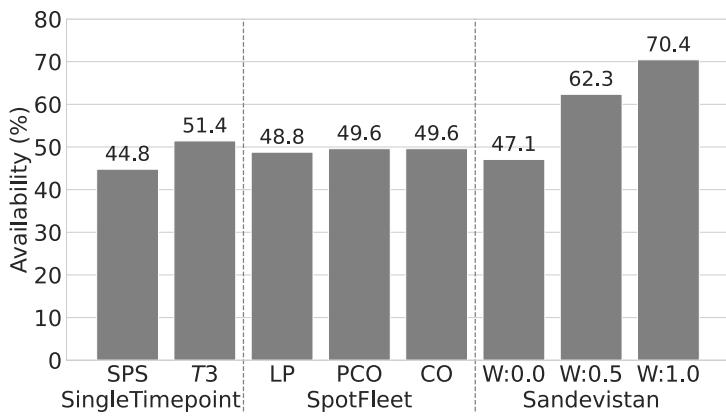
다음으로, 가용성 점수에 따른 다중 노드 스팟 인스턴스의 가용성을 시각화하기 위해 KME를 사용하여 스팟 인스턴스의 생존율을 추정했다. KME는 다음과 같이 계산된다:

$$\hat{S}(t) = \prod_{i: t_i \leq t} \left( \frac{n_i - d_i}{n_i} \right)$$

시간  $t$ 에서의 생존 함수  $\hat{S}(t)$ 는 스팟 인스턴스의 실행 시간이 시간  $t$ 를 초과할 확률로 정의된다. 여기서  $n_i$ 는 시간  $t_i$ 에 실행 중인 인스턴스 수이고,  $d_i$ 는 시간  $t_i$ 에 중단된 스팟 인스턴스 수이다. 그림 9는 다양한 가용성 점수 범위에 따른  $\hat{S}(t)$ 를 비교한다. 가로축은 시간을, 세로축은 생존율을 나타낸다. 가용성 점수를 25점 단위, 네 개의 구간으로 그룹화했다. 실선은 가용성 점수가 75 이상인 인스턴스를, 점선은 점수가 50에서 75 사이인 인스턴스를 나타내는 방식이다. 그림에서 볼 수 있듯이, 점수가 높은 그룹일수록 생존율이 높다. 가용성 점수가 25 미만인 인스턴스의 생존 시간의 중앙값은 13시간인 반면, 75 이상인



(a) Cost savings ratio



(b) Availability

그림 10: Sandevistan과 AWS SpotFleet 및 단일시점을 이용했을 때 가용성과 비용 효율성 비교

인스턴스의 생존 시간의 중앙값은 21.6시간이다. 이 결과는 제안된 가용성 점수를 기반으로 다중 노드 스팟 인스턴스를 선택하면 다중 노드 스팟 인스턴스 사용의 신뢰성을 크게 향상시킬 수 있음을 시사한다.

### 7.3 인스턴스 추천 시스템의 성능

마지막으로 질문 3에 답하여 Sandevistan 시스템 인스턴스 추천 효과를 검증한다. 비용과 가용성 측면에서 다양한 대상과 비교하여 그 성능을 평가한다.

첫 번째 비교 대상으로 AWS SpotFleet [51]을 선택했다. SpotFleet은 사용자가 시작 템플릿이나 매개변수 구성을 기반으로 스팟 인스턴스 풀을 쉽게 구성할 수 있게 해주며, 비용과 가용성의 균형을 맞추기 위해 다양한 할당 전략을 제공한다. 구체적으로 *Lowest Price*(LP), *Capacity Optimized*(CO), *Price-Capacity Optimized*(PCO)를 지원한다. SpotFleet의 지역 서비스 제약으로 인해 실험은 가장 큰 지역 중 하나로 알려진 *us-east-1*에서 수행되었다. 유사한 최적화 휴리스틱을 고려할 때 SpotFleet은 Sandevistan과 비교하기에 좋은 기준선이다. 또한 과거 데이터를 고려하지 않고 실험 시작 직전의 단일 노드 SPS와 T3 값을 사용하여 다중 노드 스팟 인스턴스를 선택하는 방식과도 성능을 비교했다. 이 경우 동일한 최고 안정성 데이터를 가진 인스턴스의 경우 가장 저렴한 것을 선택했다. Sandevistan의 경우 0.0, 0.5, 1.0 3가지 가중치를 사용했다. 인스턴스 선택 후, 인스턴스 할당 성공률을 관찰하기 위해 24시간 동안 10분 간격으로 50개의 스팟 인스턴스를 요청했다. 실험 기간 동안 스팟 및 온디맨드 가격이 큰 폭으로 변화하지 않았음을 확인하였다. 그럼 [10]는 다양한 인스턴스 선택 전략의 가용성과 비용을 비교하는 막대 그래프를 보여준다. 가로축은 인스턴스 선택 전략을, 세로축은 비용 절감(그림 [10a])과 가용성(그림 [10b])을 나타낸다.

그림 [10a]에서 비용 절감은 온디맨드 가격 대비 스팟 가격의 할인 비율로 계산된다. 결과는 Sandevistan의 가중치( $W$ )가 감소할수록 현저히 높은 비용 절감이 달성을 보여준다. 반면, SpotFleet의 최저 가격 전략으로 선택된 인스턴스는 다른 SpotFleet 전략보다 약간 높은 절감을 보여주지만, Sandevistan의  $W$ 가 0일 때보다 26.3% 낮은 절감을 보인다. 단일 시점 SPS도 모든 지역에서 가장 저렴한 인스턴스를 고려했기 때문에 높은 비용 절감을 보여주었다. 그러나 가용성보다 비용을 우선시하는 Sandevistan보다 높은 비용 절감을 달성하지는 못했다. 스팟 인스턴스의 가용성을 보여주는 그림 [10b]에서, Sandevistan은 가중치( $W$ )가 증가할수록 더 나은 가용성을 보여준다. SpotFleet의 CO와 PCO 전략은 LP 전략보다 약간 더 나은 가용성(0.8%)을 보여주었지만, Sandevistan의  $W$ 가 1일 때보다 20.6% 낮았다. 또한 CO와 PCO는 동일한 가용성을 보여주었는데, 이는 SpotFleet이 동일한 추천을 했기 때문이다. 더욱이 이들은 오직 가격에만 초점을 맞춘 LP와 유사한 성능을 보여주었고, 기대했던 수준의 가용성을 달성하지 못했다. 모든 단일 시점 전략은 Sandevistan의  $W$ 가 0.5와 1.0일 때보다 낮은 가용성을 보여주었다.

제안된 Sandevistan 시스템은 단일 지역 내에서만 인스턴스를 추천하는 AWS SpotFleet과 비교했을 때, 유사한 비용 절감을 유지하면서 가용성을 20% 이상 향상시킬 수 있음을 보여주었다. 또한 가용성이 유사할 때, Sandevistan은 25% 이상의 추가 비용 절감을 달성했다. 이러한 결과는 제안된 시스템의 높은 잠재력과 실용성을 보여준다.

## 제 8 장 관련논문

**스팟 인스턴스 데이터셋의 모델링 및 활용:** 스팟 인스턴스가 도입된 이후, 인스턴스 종료 가능성을 최소화하기 위해 스팟 인스턴스 가격 데이터셋과 중단 위험을 연관시키려는 많은 시도가 있었다 [2]–[5], [52]–[54]. 예를 들어, Ali-Eldin 등은 스팟 가격과 분석을 기반으로 한 웹 서버 배포 휴리스틱을 제안했고 [9], Son 등은 전역 지역의 GPU 스팟 인스턴스를 사용하여 DNN 훈련 작업을 수행하는 DeepSpotCloud [6]를 소개했다. 빅데이터 처리에서는 SeeSpotRun [7]이 Hadoop [55] MapReduce [56]를 개선했고, Apache Spark [57]를 위해 Flint [58] 와 Tr-Spark [59]가 제안되었다. 그러나 이전 연구들은 운영 변경 이후 구식이 된 과거 스팟 가격 데이터셋에 의존했다 [20], [21]. 본 논문은 스팟 가격 데이터셋에 의존하지 않는 가용성 데이터와 가격데이터를 모두 고려하는 방식으로 비용 효율적이고 안정적인 다중 스팟 인스턴스를 선택할 수 있게 한다.

**스팟 인스턴스 중단 분석:** Pham 등 [43], Lee [22], Kim [36]은 AWS 스팟 인스턴스에 대한 실험을 수행하여 중단 패턴을 분석했다. Azure의 경우, Yang 등 [60]이 Azure의 내부 중단 추적 데이터를 사용하여 Transformer 아키텍처 기반의 중단 예측 모델을 제안했다. GCP의 경우, Haugerud 등 [6]과 Kadupitiya 등 [62]이 스팟 인스턴스 중단을 모델링했다. 또한 Yang 등 [63]은 멀티 클라우드 환경에서 중단 실험을 수행했다. 본 논문이 제안한 접근 방식은 새로운 유형의 스팟 인스턴스 데이터셋을 활용하여 신뢰성을 개선하므로 이전 연구를 보완한다.

**다중 노드 스팟 인스턴스 활용:** 딥러닝과 빅데이터 처리와 같은 다양한 애플리케이션에서 컴퓨팅 리소스의 규모가 증가함에 따라, 중단을 처리하고 훈련 처리량을 개선하면서 훈련 비용을 줄이는 데 초점을 맞춰 여러 스팟 인스턴스 노드를 안정적으로 활용하기 위한 여러 접근 방식이 제안되었다 [30]–[34]. 본 논문은 이 연구에서 제안된 알고리즘이 이러한 이전 연구 결과의 유용성을 증가시킬 것으로 기대한다. Yang 등 [64]과 Xu 등 [65]은 온디맨드와 스팟 인스턴스를 혼합하여 비용 효율적이고 신뢰할 수 있는 클러스터를 구축하는 방법을 제안했다. Sharma 등 [66]과 Harlap 등 [67]은 스팟 인스턴스 클러스터의 비용과 신뢰성의 균형을 맞추는 프레임워크를 소개했다. 기존 연구가 내부 데이터셋이

나 스팟 가격 데이터셋에 의존하는 반면, 저자가 아는 한 이 논문은 가격 이외의 공개 데이터셋을 사용하여 대규모 스팟 인스턴스의 가용성을 정량적으로 모델링한 최초의 연구이다.

## 제 9 장 결론

본 논문에서는 대규모 다중 노드 환경에서 스팟 인스턴스에 대한 단일 노드 가용성 점수의 한계를 다루었다. 다중 노드 가용성 데이터셋을 효율적으로 수집하기 위한 새로운 휴리스틱을 제안하고, 여러 스팟 인스턴스의 안정성을 정량화하는 가용성 점수를 도입했으며, 비용과 가용성의 균형을 맞춘 스팟 인스턴스를 추천하는 시스템을 개발했다. 광범위한 실제 실험을 통해 제안된 시스템이 AWS 의 Spot Fleet과 같은 기존 방식에 비해 스팟 인스턴스의 신뢰성을 최대 20.6% 향상시킴을 입증했다. 또한 수집된 다중 노드 가용성 데이터셋을 웹 서비스를 통해 공개적으로 이용 가능하게 하여, 사용자들이 스팟 인스턴스를 배포할 때 더 정보에 입각한 결정을 내릴 수 있게 했다.

향후 연구는 Microsoft Azure, GCP와 같은 추가 클라우드 환경을 처리하도록 추천 시스템을 확장하고, DNN 훈련, 추론, 빅데이터 처리와 같은 실제 워크로드에 제안된 알고리즘을 적용하는 데 초점을 맞출 것이다.

## 참고 문헌

- [1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, “Deconstructing amazon ec2 spot instance pricing,” *ACM Trans. Econ. Comput.*, vol. 1, no. 3, 2013, ISSN: 2167-8375. DOI: 10 . 1145 / 2509413 . 2509416 [Online]. Available: <https://doi.org/10.1145/2509413.2509416>.
- [2] D. Movsowitz Davidow, O. Agmon Ben-Yehuda, and O. Dunkelman, “Deconstructing alibaba cloud’s preemptible instance pricing,” in *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’23, Orlando, FL, USA: Association for Computing Machinery, 2023, 253–265, ISBN: 9798400701559. [Online]. Available: <https://dl.acm.org/pdf/10.1145/3588195.3593001>.
- [3] B. Javadi, R. K. Thulasiram, and R. Buyya, “Characterizing spot price dynamics in public cloud environments,” *Future Generation Computer Systems*, vol. 29, no. 4, pp. 988–999, 2013, Special Section: Utility and Cloud Computing, ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2012.06.012> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X12001483>.
- [4] C. Wang, Q. Liang, and B. Urgaonkar, “An empirical analysis of amazon ec2 spot instance features affecting cost-effective resource procurement,” in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE ’17, L’Aquila, Italy: Association for Computing Machinery, 2017, 63–74, ISBN: 9781450344043. DOI: 10 . 1145 / 3030207 . 3030210 [Online]. Available: <https://doi.org/10.1145/3030207.3030210>.
- [5] N. Ekwe-Ekwe and A. Barker, “Location, location, location: Exploring amazon ec2 spot instance pricing across geographical regions,” in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid*

*Computing (CCGRID)*, 2018, pp. 370–373. DOI: [10.1109/CCGRID.2018.00059](https://doi.org/10.1109/CCGRID.2018.00059).

- [6] K. Lee and M. Son, “Deepspotcloud: Leveraging cross-region gpu spot instances for deep learning,” in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 98–105. DOI: [10.1109/CLOUD.2017.21](https://doi.org/10.1109/CLOUD.2017.21).
- [7] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, “See spot run: Using spot instances for MapReduce workflows,” in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, Boston, MA: USENIX Association, Jun. 2010. [Online]. Available: <https://www.usenix.org/conference/hotcloud-10/see-spot-run-using-spot-instances-mapreduce-workflows>.
- [8] P. Sharma, T. Guo, X. He, D. Irwin, and P. Shenoy, “Flint: Batch-interactive data-intensive processing on transient servers,” in *Proceedings of the Eleventh European Conference on Computer Systems*, ser. EuroSys ’16, London, United Kingdom: Association for Computing Machinery, 2016, ISBN: 9781450342407. DOI: [10.1145/2901318.2901319](https://doi.org/10.1145/2901318.2901319). [Online]. Available: <https://doi.org/10.1145/2901318.2901319>.
- [9] A. Ali-Eldin, J. Westin, B. Wang, P. Sharma, and P. Shenoy, “Spotweb: Running latency-sensitive distributed web services on transient cloud servers,” in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’19, Phoenix, AZ, USA: Association for Computing Machinery, 2019, 1–12, ISBN: 9781450366700. DOI: [10.1145/3307681.3325397](https://doi.org/10.1145/3307681.3325397). [Online]. Available: <https://doi.org/10.1145/3307681.3325397>
- [10] X. He, P. Shenoy, R. Sitaraman, and D. Irwin, “Cutting the cost of hosting online services using cloud spot markets,” in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’15, Portland, Oregon, USA: Association for

- Computing Machinery, 2015, 207–218, ISBN: 9781450335508. DOI: [10 . 1145 / 2749246 . 2749275](https://doi.org/10.1145/2749246.2749275). [Online]. Available: <https://doi.org/10.1145/2749246.2749275>.
- [11] I. Menache, O. Shamir, and N. Jain, “On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud,” in *11th International Conference on Autonomic Computing (ICAC '14)*, Philadelphia, PA: USENIX Association, Jun. 2014, pp. 177–187, ISBN: 978-1-931971-11-9. [Online]. Available: <https://www.usenix.org/conference/icac14/technical-sessions/presentation/menache>.
- [12] S. Subramanya, T. Guo, P. Sharma, D. Irwin, and P. Shenoy, “Spoton: A batch computing service for the spot market,” in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC ’15, Kohala Coast, Hawaii: Association for Computing Machinery, 2015, 329–341, ISBN: 9781450336512. DOI: [10 . 1145 / 2806777 . 2806851](https://doi.org/10.1145/2806777.2806851). [Online]. Available: <https://doi.org/10.1145/2806777.2806851>.
- [13] P. Varshney and Y. Simmhan, “Autobot: Resilient and cost-effective scheduling of a bag of tasks on spot vms,” *IEEE Transactions on Parallel & Distributed Systems*, vol. 30, no. 07, pp. 1512–1527, 2019, ISSN: 1558-2183. DOI: [10 . 1109/TPDS.2018.2889851](https://doi.org/10.1109/TPDS.2018.2889851).
- [14] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, “How to bid the cloud,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM ’15, London, United Kingdom: Association for Computing Machinery, 2015, 71–84, ISBN: 9781450335423. DOI: [10 . 1145 / 2785956 . 2787473](https://doi.org/10.1145/2785956.2787473). [Online]. Available: <https://doi.org/10.1145/2785956.2787473>.
- [15] P. Sharma, D. Irwin, and P. Shenoy, “How not to bid the cloud,” in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '16)*, Denver, CO: USENIX Association, Jun. 2016. [Online]. Available: <https://doi.org/10.1145/2785956.2787473>.

---

//www.usenix.org/conference/hotcloud16/workshop-program/presentation/sharma.

- [16] S. Alkharif, K. Lee, and H. Kim, “Time-series analysis for price prediction of opportunistic cloud computing resources,” in *Proceedings of the 7th International Conference on Emerging Databases*, W. Lee, W. Choi, S. Jung, and M. Song, Eds., Singapore: Springer Singapore, 2018, pp. 221–229, ISBN: 978-981-10-6520-0.
- [17] V. Khandelwal, A. Chaturvedi, and C. P. Gupta, “Amazon ec2 spot price prediction using regression random forests,” *IEEE Transactions on Cloud Computing*, pp. 1–1, 2017. DOI: [10.1109/TCC.2017.2780159](https://doi.org/10.1109/TCC.2017.2780159).
- [18] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, and M. Chiang, “Learning cloud dynamics to optimize spot instance bidding strategies,” in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2762–2770. DOI: [10.1109/INFocom.2018.8486291](https://doi.org/10.1109/INFocom.2018.8486291).
- [19] J. Fabra, J. Ezpeleta, and P. Álvarez, “Reducing the price of resource provisioning using ec2 spot instances with prediction models,” *Future Generation Computer Systems*, vol. 96, pp. 348–367, 2019, ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2019.01.025> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X1831166X>.
- [20] M. Baughman, S. Caton, C. Haas, *et al.*, “Deconstructing the 2017 changes to aws spot market pricing,” in *Proceedings of the 10th Workshop on Scientific Cloud Computing*, ser. ScienceCloud ’19, Phoenix, AZ, USA: Association for Computing Machinery, 2019, 19–26, ISBN: 9781450367585. DOI: [10.1145/3322795.3331465](https://doi.org/10.1145/3322795.3331465) [Online]. Available: <https://doi.org/10.1145/3322795.3331465>.
- [21] D. Irwin, P. Shenoy, P. Ambati, P. Sharma, S. Shastri, and A. Ali-Eldin, “The price is (not) right: Reflections on pricing for transient cloud servers,” in *2019 28th International Conference on Computer*

*Communication and Networks (ICCCN)*, 2019, pp. 1–9. DOI: [10.1109/ICCCN.2019.8846933](https://doi.org/10.1109/ICCCN.2019.8846933).

- [22] S. Lee, J. Hwang, and K. Lee, “Spotlake: Diverse spot instance dataset archive service,” in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 242–255. DOI: [10.1109/IISWC55918.2022.00029](https://doi.org/10.1109/IISWC55918.2022.00029). [Online]. Available: <https://doi.ieee.org/10.1109/IISWC55918.2022.00029>.
- [23] S. Tang, J. Yuan, and X.-Y. Li, “Towards optimal bidding strategy for amazon ec2 cloud spot instance,” in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 91–98. DOI: [10.1109/CLOUD.2012.134](https://doi.org/10.1109/CLOUD.2012.134).
- [24] azure cloud, *Azure spot virtual machines*, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/virtual-machines/spot-vms/>.
- [25] alibaba cloud, *Alibaba preemptible instance*, 2024. [Online]. Available: <https://www.alibabacloud.com/help/en/ecs/user-guide/overview-4/>.
- [26] A. W. is New, *Introducing amazon ec2 spot placement score*, 2021. [Online]. Available: <https://aws.amazon.com/about-aws/whats-new/2021/10/amazon-ec2-spot-placement-score/>.
- [27] Azure, *Az compute-diagnostic spot-placement-recommender*, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/cli/azure/compute-diagnostic/spot-placement-recommender?view=azure-cli-latest>.
- [28] R. Yazdani Aminabadi, S. Rajbhandari, M. Zhang, *et al.*, “Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale,” Microsoft, Tech. Rep. MSR-TR-2022-21, 2022.

- [Online]. Available: <https://www.microsoft.com/en-us/research/publication/deepspeed-inference-enabling-efficient-inference-of-transformer-models-at-unprecedented-scale/>
- [29] J. Sampé, M. Sánchez-Artigas, G. Vernik, I. Yehekzel, and P. García-López, “Outsourcing data processing jobs with lithops,” *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 1026–1037, 2023. DOI: [10.1109/TCC.2021.3129000](https://doi.org/10.1109/TCC.2021.3129000).
- [30] J. Thorpe, P. Zhao, J. Eyolfson, *et al.*, “Bamboo: Making preemptible instances resilient for affordable training of large DNNs,” in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, Boston, MA: USENIX Association, Apr. 2023, pp. 497–513, ISBN: 978-1-939133-33-5. [Online]. Available: <https://www.usenix.org/conference/nsdi23/presentation/thorpe>.
- [31] J. Duan, Z. Song, X. Miao, *et al.*, “Parcae: Proactive, Liveput-Optimized DNN training on preemptible instances,” in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1121–1139, ISBN: 978-1-939133-39-7. [Online]. Available: <https://www.usenix.org/conference/nsdi24/presentation/duan>.
- [32] I. Jang, Z. Yang, Z. Zhang, X. Jin, and M. Chowdhury, “Oobleck: Resilient distributed training of large models using pipeline templates,” in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP ’23, Koblenz, Germany: Association for Computing Machinery, 2023, 382–395, ISBN: 9798400702297. DOI: [10.1145/3600006.3613152](https://doi.org/10.1145/3600006.3613152). [Online]. Available: <https://doi.org/10.1145/3600006.3613152>
- [33] S. Athlur, N. Saran, M. Sivathanu, R. Ramjee, and N. Kwatra, “Varuna: Scalable, low-cost training of massive deep learning models,” in *Proceedings of the Seventeenth European Conference on Computer Systems*, ser. EuroSys ’22, Rennes, France: Association for Computing

- Machinery, 2022, 472–487, ISBN: 9781450391627. DOI: 10 . 1145 / 3492321 . 3519584, [Online]. Available: <https://doi.org/10.1145/3492321.3519584>,
- [34] Y. Kim, K. Kim, Y. Cho, *et al.*, “Deepvm: Integrating spot and on-demand vms for cost-efficient deep learning clusters in the cloud,” in *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2024, pp. 227–235. DOI: 10 . 1109/CCGrid59990 . 2024 . 00034,
- [35] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krantz, “See spot run: Using spot instances for {mapreduce} workflows,” in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.
- [36] K. Kim and K. Lee, “Making cloud spot instance interruption events visible,” in *Proceedings of the ACM on Web Conference 2024*, ser. WWW ’24, Singapore, Singapore: Association for Computing Machinery, 2024, 2998–3009, ISBN: 9798400701719. DOI: 10 . 1145 / 3589334 . 3645548, [Online]. Available: <https://doi.org/10.1145/3589334.3645548>,
- [37] S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019, ISBN: 0470183527.
- [38] S. Raschka, “About feature scaling and normalization and the effect of standardization for machine learning algorithms,” Jul. 2014. DOI: 10 . 13140/2 . 1 . 4245 . 1849,
- [39] S. Narayan, “The generalized sigmoid activation function: Competitive supervised learning,” *Information Sciences*, vol. 99, no. 1, pp. 69–82, 1997, ISSN: 0020-0255. DOI: [https://doi.org/10.1016/S0020-0255\(96\)00200-9](https://doi.org/10.1016/S0020-0255(96)00200-9), [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025596002009>,
- [40] A. Karalič and I. Bratko, “First order regression,” *Machine learning*, vol. 26, pp. 147–176, 1997.

- [41] J. M. Hellerstein, J. M. Faleiro, J. Gonzalez, *et al.*, “Serverless computing: One step forward, two steps back,” in *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*, [www.cidrdb.org](http://cidrdb.org/cidr2019/papers/p119-hellerstein-cidr19.pdf), 2019. [Online]. Available: <http://cidrdb.org/cidr2019/papers/p119-hellerstein-cidr19.pdf>
- [42] J. Kadupitige, V. Jadhao, and P. Sharma, “Modeling the temporally constrained preemptions of transient cloud vms,” in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’20, Stockholm, Sweden: Association for Computing Machinery, 2020, 41–52, ISBN: 9781450370523. DOI: [10.1145/3369583.3392671](https://doi.org/10.1145/3369583.3392671). [Online]. Available: <https://doi.org/10.1145/3369583.3392671>
- [43] T.-P. Pham, S. Ristov, and T. Fahringer, “Performance and behavior characterization of amazon ec2 spot instances,” in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 73–81. DOI: [10.1109/CLOUD.2018.00017](https://doi.org/10.1109/CLOUD.2018.00017)
- [44] Z. Wu, W.-L. Chiang, Z. Mao, *et al.*, “Can’t be late: Optimizing spot instance savings under deadlines,” in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, Santa Clara, CA: USENIX Association, Apr. 2024, pp. 185–203, ISBN: 978-1-939133-39-7. [Online]. Available: <https://www.usenix.org/conference/nsdi24/presentation/wu-zhanghao>
- [45] E. L. Kaplan and P. Meier, “Nonparametric estimation from incomplete observations,” *Journal of the American Statistical Association*, vol. 53, no. 282, pp. 457–481, 1958. DOI: [10.1080/01621459.1958.10501452](https://doi.org/10.1080/01621459.1958.10501452). eprint: <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501452>. [Online]. Available: <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501452>

<https://www.tandfonline.com/doi/abs/10.1080/01621459.1958.10501452>.

- [46] D. R. Cox, “Regression models and life-tables,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.
- [47] D. A. Ali and A. M. Hussein, “Analysis of cox proportional hazard model for dropout students in university: Case study from simad university,” *Journal of Applied Research in Higher Education*, vol. 16, no. 3, pp. 820–830, 2024.
- [48] K. N. Chi, T Kheoh, C. J. Ryan, *et al.*, “A prognostic index model for predicting overall survival in patients with metastatic castration-resistant prostate cancer treated with abiraterone acetate after docetaxel,” *Annals of Oncology*, vol. 27, no. 3, pp. 454–460, 2016.
- [49] M. Okuda-Arai, S. Mori, F. Takano, *et al.*, “Impact of glaucoma medications on subsequent schlemm’s canal surgery outcome: Cox proportional hazard model and propensity score-matched analysis,” *Acta Ophthalmologica*, vol. 102, no. 2, e178–e184, 2024.
- [50] D. Conniffe, “Expected maximum log likelihood estimation,” *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 36, no. 4, pp. 317–329, 1987, ISSN: 00390526, 14679884. [Online]. Available: <http://www.jstor.org/stable/2348828> (visited on 10/11/2024).
- [51] aws, *Ec2 fleet and spot fleet*, 2024. [Online]. Available: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Fleets.html>.
- [52] R. Wolski, J. Brevik, R. Chard, and K. Chard, “Probabilistic guarantees of execution duration for amazon spot instances,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’17, Denver, Colorado: Association for Computing Machinery, 2017, ISBN: 9781450351140. DOI:

- [10 . 1145 / 3126908 . 3126953]. [Online]. Available: <https://doi.org/10.1145/3126908.3126953>.
- [53] G. Portella, G. N. Rodrigues, E. Nakano, and A. C. Melo, “Statistical analysis of amazon ec2 cloud pricing models,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 18, e4451, 2019, e4451 cpe.4451. DOI: <https://doi.org/10.1002/cpe.4451>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4451>
- [54] A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, and M. Schulz, “Exploiting redundancy for cost-effective, time-constrained execution of hpc applications on amazon ec2,” in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’14, Vancouver, BC, Canada: Association for Computing Machinery, 2014, 279–290, ISBN: 9781450327497. DOI: 10 . 1145 / 2600212 . 2600226. [Online]. Available: <https://doi.org/10.1145/2600212.2600226>.
- [55] A. S. Foundation, *Apache hadoop*, 2004. [Online]. Available: <http://hadoop.apache.org/>.
- [56] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, ser. OSDI’04, San Francisco, CA: USENIX Association, 2004, pp. 10–10. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1251254.1251264>.
- [57] M. Zaharia, M. Chowdhury, T. Das, *et al.*, “Resilient distributed datasets: A Fault-Tolerant abstraction for In-Memory cluster computing,” in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, San Jose, CA: USENIX Association, Apr. 2012, pp. 15–28, ISBN: 978-931971-92-8. [Online]. Available: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia>.

- [58] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache flink: Stream and batch processing in a single engine,” *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.
- [59] Y. Yan, Y. Gao, Y. Chen, Z. Guo, B. Chen, and T. Moscibroda, “Tr-spark: Transient computing for big data analytics,” in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, ser. SoCC ’16, Santa Clara, CA, USA: Association for Computing Machinery, 2016, 484–496, ISBN: 9781450345255. DOI: [10.1145/2987550.2987576](https://doi.org/10.1145/2987550.2987576). [Online]. Available: <https://doi.org/10.1145/2987550.2987576>
- [60] F. Yang, B. Pang, J. Zhang, *et al.*, “Spot virtual machine eviction prediction in microsoft cloud,” in *Companion Proceedings of the Web Conference 2022*, ser. WWW ’22, Virtual Event, Lyon, France: Association for Computing Machinery, 2022, 152–156, ISBN: 9781450391306. DOI: [10.1145/3487553.3524229](https://doi.org/10.1145/3487553.3524229). [Online]. Available: <https://doi.org/10.1145/3487553.3524229>
- [61] H. Haugerud, J. Krüger Svensson, and A. Yazidi, “Autonomous provisioning of preemptive instances in google cloud for maximum performance per dollar,” in *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)*, 2020, pp. 1–8. DOI: [10.1109/CloudTech49835.2020.9365879](https://doi.org/10.1109/CloudTech49835.2020.9365879).
- [62] J. Kadupitiya, V. Jadhao, and P. Sharma, “Scispot: Scientific computing on temporally constrained cloud preemptible vms,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3575–3588, 2022. DOI: [10.1109/TPDS.2022.3157272](https://doi.org/10.1109/TPDS.2022.3157272).
- [63] Z. Yang, Z. Wu, M. Luo, *et al.*, “SkyPilot: An intercloud broker for sky computing,” in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, Boston, MA: USENIX Association, Apr. 2023, pp. 437–455, ISBN: 978-1-939133-33-5. [Online]. Available:

<https://www.usenix.org/conference/nsdi23/presentation/yang-zongheng>

- [64] F. Yang, L. Wang, Z. Xu, *et al.*, “Snape: Reliable and low-cost computing with mixture of spot and on-demand vms,” in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS 2023, Vancouver, BC, Canada: Association for Computing Machinery, 2023, 631–643, ISBN: 9781450399180. DOI: [10.1145/3582016.3582028](https://doi.org/10.1145/3582016.3582028) [Online]. Available: <https://doi.org/10.1145/3582016.3582028>
- [65] Z. Xu, C. Stewart, N. Deng, and X. Wang, “Blending on-demand and spot instances to lower costs for in-memory storage,” in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9. DOI: [10.1109/INFOCOM.2016.7524348](https://doi.org/10.1109/INFOCOM.2016.7524348).
- [66] P. Sharma, D. Irwin, and P. Shenoy, “Portfolio-driven resource management for transient cloud servers,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 1–23, 2017.
- [67] A. Harlap, A. Chung, A. Tumanov, G. R. Ganger, and P. B. Gibbons, “Tributary: Spot-dancing for elastic services with latency SLOs,” in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, Boston, MA: USENIX Association, Jul. 2018, pp. 1–14, ISBN: 978-1-931971-44-7. [Online]. Available: <https://www.usenix.org/conference/atc18/presentation/harlap>.

## 영문 요약

Cloud providers offer discounted spot instances to maximize surplus resource utilization, but these come with the risk of sudden interruption. Traditional pricing datasets have been used to predict this risk, but recent policy changes have weakened their effectiveness. To encourage spot instance usage, cloud vendors provide instant availability dataset to help users mitigate this risk. While existing research utilizing this data has proposed methods to reduce the likelihood of interruptions, these studies have primarily focused on single-node instances, neglecting the stability of multi-node environments that are widely adopted for recent cloud workloads.

In this paper, we propose Sandevistan, a system that recommends optimal multi-node spot instances by considering both multi-node availability and price data. Sandevistan overcomes various query limitations to collect huge multi-node availability data, and through thorough analysis, it provides a methodology for recommending cost-efficient and reliable multi-node spot instances. To evaluate how well the proposed methodology reflects multi-node availability and cost efficiency, we conducted extensive real-world interruption experiments. The results show that, depending on the recommendation method, Sandevistan can achieve 20.8% better stability and 26.3% higher cost savings compared to public cloud services.

keywords: Cloud computing, Spot instances, Computing resources, Optimal multi node spot instances, Web log analysis

**표 1:** 쿼리 표기법

Notation	Definition
$p$	Query period
$T_{max}$	Maximum number of target nodes to query SPS
$T_{min}$	Minimum number of target nodes to query SPS
$T_c$	The number of nodes to query in the current iteration
$T_s$	The number of nodes to increase in every other query
$T_3$	Maximum number of nodes whose SPS value is three
$T_2$	Maximum number of nodes whose SPS value is two
$\alpha$	Hyperparameter to determine the initial number of nodes to send SPS query
$T_{high}$	The upper bound of binary search space
$T_{low}$	The lower bound of binary search space
$T_{curr}$	Target capacity to send query
$T_{3_{prev}}$	$T_3$ value in the previous data collection period
$T_S$	Target SPS score to track for transition point, 2 or 3
$S_t$	SPS score of the queried target capacity, $t$
$e$	Threshold for terminating the search

## $T_3$ 추적 알고리즘

표 1은 본 논문에서 사용된 다양한 표기법과 함께 이진 검색 기반의  $T_3$  검색 및 최적화 기법의 알고리즘 설명에서 새롭게 도입된 주요 표기법을 요약한다.

알고리즘 1은 기본 이진 검색 방법에 *Caching*과 *Early Stop* 기법을 적용하여  $T_3$ 에 해당하는 노드 수를 결정하는 의사 코드를 제시한다.  $T_2$ 에 대해서도 유사한 쿼리 과정이 수행된다. 알고리즘의 입력은 인스턴스 유형과 AZ를 포함하며, 알고리즘은 크게 *Load cached value* 단계(1-14줄)와 *Search with early stop* 단계(15-23줄)로 나눌 수 있다.

*Load cached value* 단계는 이전 라운드의 캐시된  $T_3$  값을 사용하여 검색 범위를 설정하는 것을 포함한다. 입력된 인스턴스 유형과 AZ를 사용하여 이전 쿼리 반복의  $T_3$  값을 검색하고  $T_{3_{prev}}$ 에 할당한다(3줄). 4-8줄에서  $T_{3_{prev}}$ 를 검색 범위의 중간점과 비교하고, 쿼리의 대상 용량인  $T_{curr}$ 를 그에 따라 업데이트한다. 여기서  $\alpha$ 는 하이퍼파라미터로,  $\alpha$  값이 클수록 현재  $T_3$  값이 이전 값에 비해 더 큰 변화를 가정한다.  $T_{3_{prev}}$ 가 검색 범위의 중간점보다 작으면  $T_{curr}$

---

**Algorithm 1** Find- $T3$ 

---

```
1: Input : InstanceType, AZ
2: Output : Change Point of SPS value
3:  $T3_{prev} \leftarrow$  Get  $T3$  from the prior period collection       $\triangleright$  Load cached value
4: if  $T3_{prev} < \lfloor(T_{min} + T_{max})/2\rfloor$  then
5:    $T_{curr} \leftarrow T3_{prev} + \alpha$ 
6: else
7:    $T_{curr} \leftarrow T3_{prev} - \alpha$ 
8: end if
9:  $S_t \leftarrow$  Get  $T_{curr}$ 's SPS score using API call
10: if  $S_t \geq TS$  then
11:    $T_{low} = \max(T_{min}, T_{curr} + 1)$ 
12: else
13:    $T_{high} = \min(T_{max}, T_{curr})$ 
14: end if
15: while  $e < T_{high} - T_{low}$  do                                 $\triangleright$  Search with early stop
16:    $T_{curr} \leftarrow \lfloor(T_{high} + T_{low})/2\rfloor$ 
17:    $S_t \leftarrow$  Get  $T_{curr}$ 's SPS score using API call
18:   if  $S_t \geq TS$  then
19:      $T_{low} = T_{curr} + 1$ 
20:   else
21:      $T_{high} = T_{curr}$ 
22:   end if
23: end while
```

---

를  $+\alpha$ 만큼 이동시키고, 크면  $-\alpha$ 만큼 이동시킨다(4-7줄). 그 다음  $T_{curr}$ 의 현재 SPS 값인  $S_t$ 를 확인하고 검색 범위를 그에 따라 업데이트한다(9-14줄). 여기서  $TS$ 는 대상 노드 수를 변경할 때 추적하는 SPS 점수를 나타내며, 예를 들어  $T3$ 를 검색할 때는  $TS = 3$ 이다.  $S_t$ 가  $TS$  이상이면 검색 범위의 하한을  $T_{curr} + 1$ 로 업데이트하고, 그렇지 않으면 상한을 업데이트한다. 이 *load cached value* 단계는 초기 검색 범위를 효율적으로 설정하여 후속 검색 단계에서 더 빠른 검색을 준비한다.

*Search with early stop* 단계는 초기화된 범위 내에서 SPS 값이 변경되는 노드 수를 찾기 위해 이진 검색을 수행하는 단계이다. 검색 범위( $T_{high} - T_{low}$ )가  $e$ 보다 큰 동안 검색이 계속된다(15줄). 여기서  $e$ 는 검색을 중지하는 임계값으로, 허용 가능한 오차 범위를 나타내며 비음의 정수이다.  $e$  값이 클수록 검색을 더

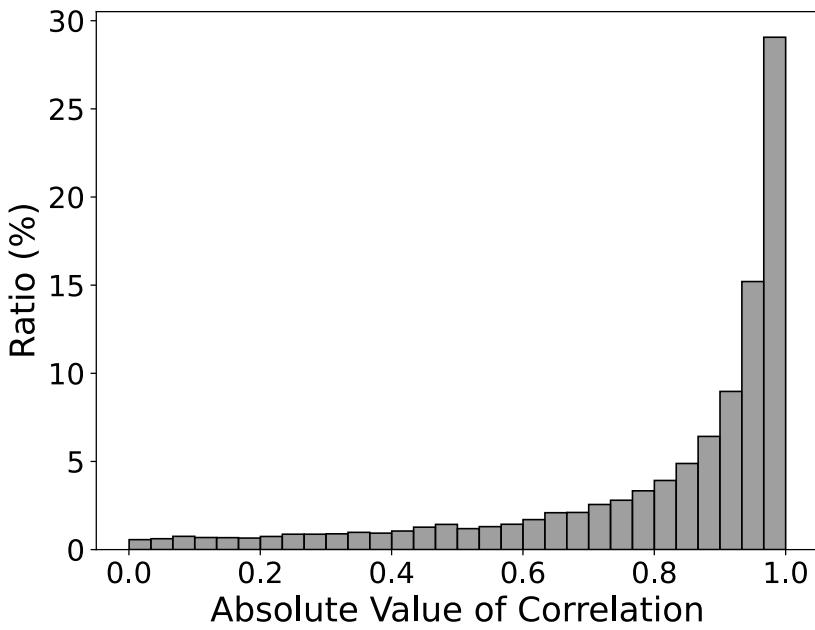


그림 11:  $T_3$ 와  $T_2$ 로 계산한 안정성 점수의 상관 계수의 절대값 분포.

일찍 종료할 수 있는 오차 범위가 증가한다. 각 반복에서 검색 범위의 중간점인  $T_{curr}$ 를 계산하고 SPS 값을 확인한다. SPS 값에 따라 검색 범위의 상한 또는 하한을 업데이트한다(17-23줄).

제안된 알고리즘은 과거 값을 사용하여 검색 범위를 초기화하고 이진 검색을 통해 SPS 값의 변곡점( $T_3$ ,  $T_2$ )을 효율적으로 식별한다. 히스토리 데이터를 활용하여 검색 범위를 좁히고 중지 임계값을 도입함으로써 이진 검색 반복 횟수를 줄여 전체적인 쿼리 오버헤드를 감소시킨다.

## $T_3$ 만을 이용한 가용성 점수 계산의 타당성

본 논문에서는 가용성 점수를 계산하기 위해  $T_3$  값을 사용했다. 그러나  $T_3$ 만을 기반으로 한 계산이 충분한지, 아니면  $T_2$  값도 포함해야 하는지 판단하기 위해,  $T_3$ 와  $T_2$  데이터셋을 모두 사용하여 가용성 점수를 계산하고 그 상관관계를 분석했다. 2024년 9월 4일부터 9월 11일까지,  $T_3$ 와  $T_2$  데이터셋을 각각 사용하여

일일 가용성 점수를 계산하고, 두 데이터셋에서 도출된 점수 간의 상관관계를 분석했다.

그림 11은 계산된 가용성 점수 간의 상관관계 절대값 분포를 보여준다. 그림의 가로축은 상관계수의 절대값을, 세로축은 각 범위의 비율을 나타낸다. 주목할 만한 점은 상관계수의 절대값이 1.0에 가까운 경우가 약 30%를 차지한다는 것인데, 이는 T3와 T2에서 계산된 가용성 점수 사이에 강한 선형 관계가 있음을 시사한다. 또한, 상관계수의 절대값이 0.6 미만인 경우는 상대적으로 적은 비율을 차지하며, 이는 T3에서 계산된 가용성 점수가 T2의 데이터 패턴을 충분히 반영하고 있음을 나타낸다. 따라서 본 논문에서는 T3만을 기반으로 한 가용성 점수가 스팟 인스턴스의 안정성을 예측하는 데 충분하다고 결론 내린다.