



기초 캐시 전략부터 알아보는 ElastiCache 입문

가동식

인하대학교 컴퓨터공학과

Introduce



인하대학교 ACC
가동식입니다.

어제의 나보다 더 성장한
오늘의 나를 위해 노력합니다.

Info

EMAIL: ret0422@gmail.com

Github: <https://github.com/Nesquito>

LinkedIn: www.linkedin.com/in/Nesquito

Content

Cache
Redis
ElastiCache

Cache



Cache란?

데이터나 값을 미리 복사해두는 임시 저장공간
원래 소스보다 더 빠르고 효율적으로 접근 가능

CPU L1, L2, L3

HTTP Cache

DRAM, HDD Cache

Server Cache

CDN

Proxy Cache

Cache



Cache란?

데이터나 값을 미리 복사해두는 임시 저장공간
원래 소스보다 더 빠르고 효율적으로 접근 가능

CPU L1, L2, L3

HTTP Cache

DRAM, HDD Cache

Server Cache

CDN

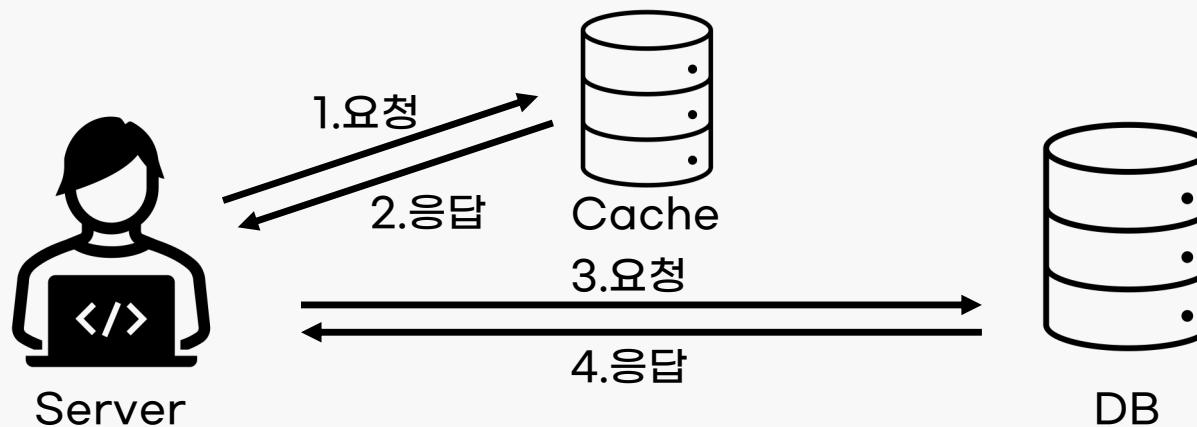
Proxy Cache

Cache



Cache란?

데이터나 값을 미리 복사해두는 임시 저장공간
원래 소스보다 더 빠르고 효율적으로 접근 가능



 **Cache**

왜 사용할까?

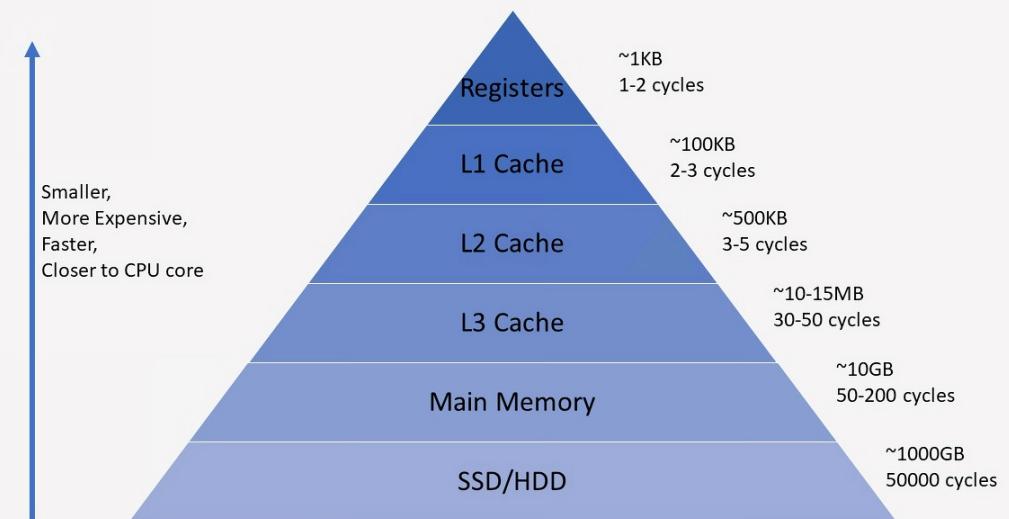
Cache



왜 사용할까?

응답 속도

- 원 데이터를 가져오는 것보다 빠르게 응답 가능



출처: <https://velog.io/@hyejin4169>

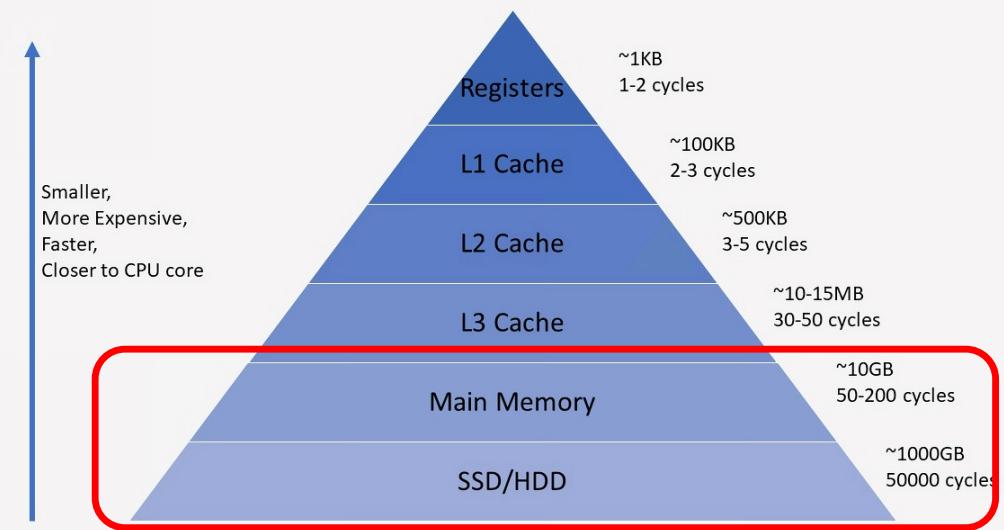
Cache



왜 사용할까?

응답 속도

- 원 데이터를 가져오는 것보다 빠르게 응답 가능



출처: <https://velog.io/@hyejin4169>

Cache



왜 사용할까?

응답 속도

- 원 데이터를 가져오는 것보다 빠르게 응답 가능

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	3 ns	10 s
Level 3 cache access	10 ns	33 s
Main memory access (DRAM, from CPU)	100 ns	6 min
Solid-state disk I/O (flash memory)	10–100 µs	9–90 hours

출처: <https://velog.io/@0xf4d3c0d3/Systems-Performance-2-Methodologies>

Cache



왜 사용할까?

응답 속도

- 원 데이터를 가져오는 것보다 빠르게 응답 가능

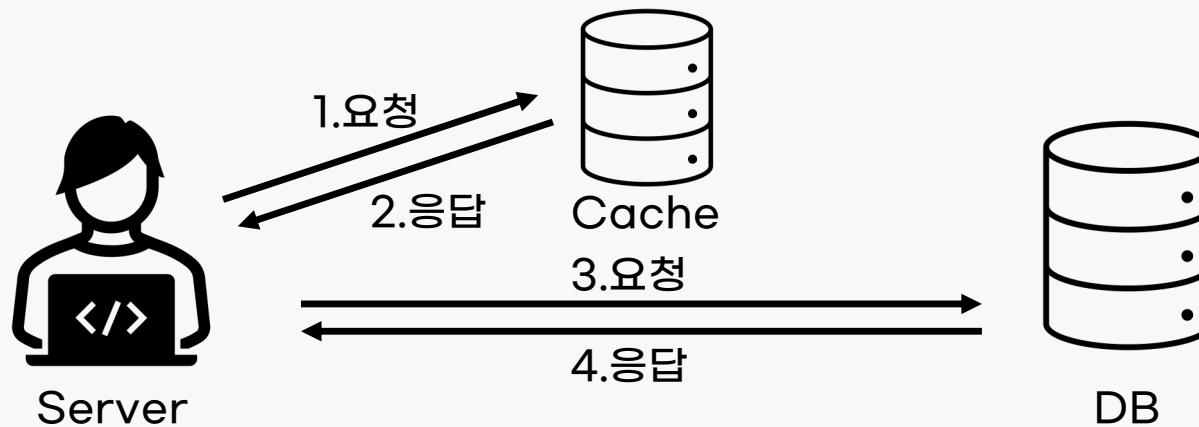
Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	3 ns	10 s
Level 3 cache access	10 ns	33 s
Main memory access (DRAM, from CPU)	100 ns	6 min
Solid-state disk I/O (flash memory)	10–100 μ s	9–90 hours

출처: <https://velog.io/@0xf4d3c0d3/Systems-Performance-2-Methodologies>

Cache



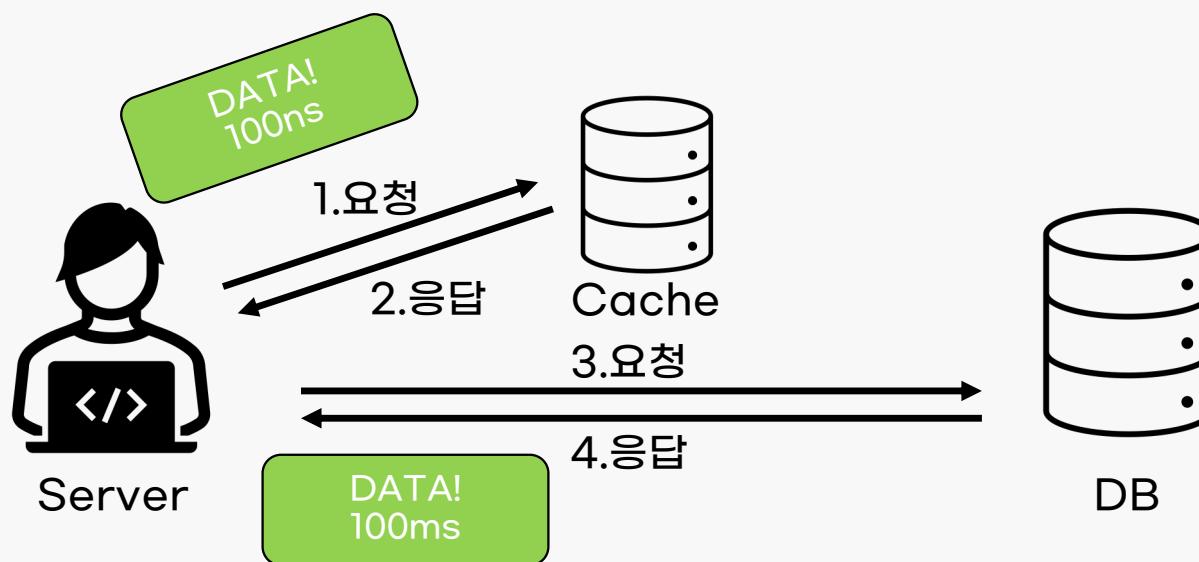
왜 사용할까?



Cache



왜 사용할까?



Cache



왜 사용할까?

반복 연산

- 반복되는 연산을 줄이는데 이용 가능

Factorial 의 계산

- $10! = 10 * 9! = 10 * 9 * 8! = 10 * 9 * 8 * 7!$
- $19! = 19 * 18 * 17 * 16 * 15 * 14 * 13 * 12 * 11 * 10!$
- 팩토리얼 숫자가 크다면?
 - 20880!, 20881! 를 매번 계산해야 한다면?

출처: <https://www.youtube.com/watch?v=mPB2CZiAkKM>

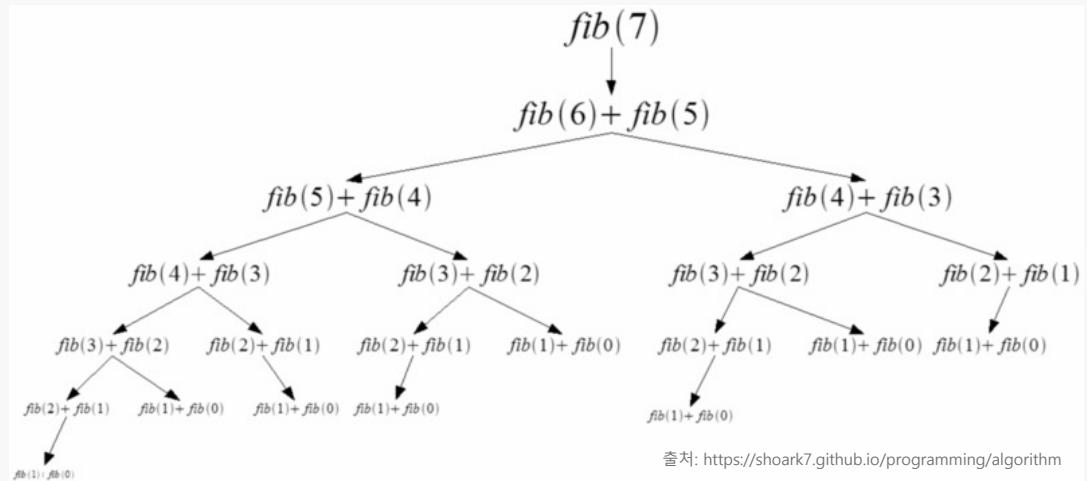
Cache



왜 사용할까?

반복 연산

- 반복되는 연산을 줄이는데 이용 가능



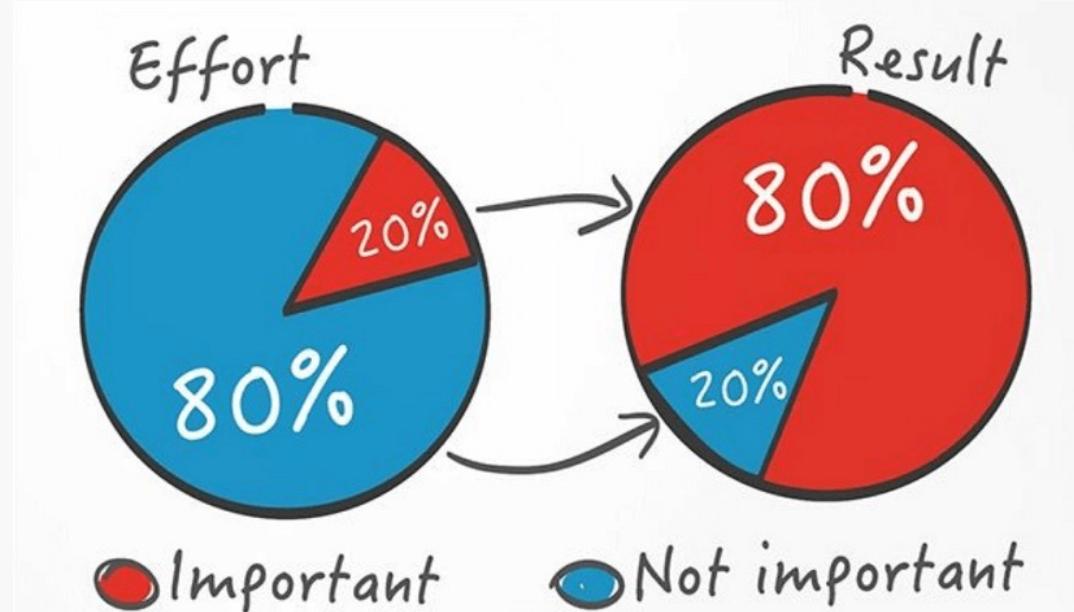
Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지



출처: <https://psycho-tests.com/blog/pareto-law>

Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지

Hit Rate 80%

Miss Rate 20%

Cache 접근시간 0.1 μ s

DB 접근시간 100 μ s

Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지

Hit Rate 80%

Miss Rate 20%

Cache 접근시간 0.1 μ s

DB 접근시간 100 μ s

Hit Rate x Cache 접근시간

+ **Miss Rate x (Cache 접근시간 + DB 접근시간)**

Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지

Hit Rate 80%

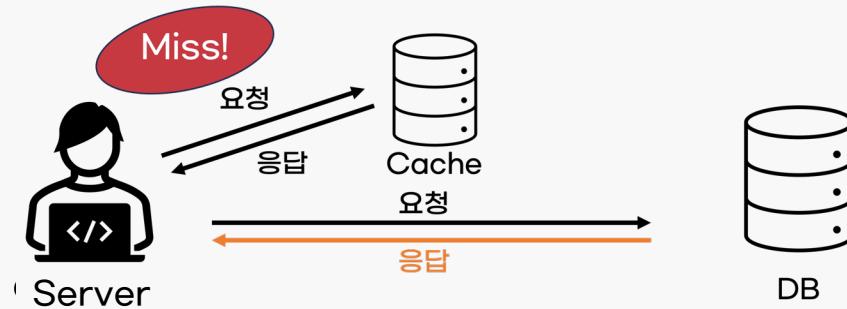
Miss Rate 20%

Cache 접근시간 0.1 μ s

DB 접근시간 100 μ s

Hit Rate x Cache 접근시간

+ **Miss Rate x (Cache 접근시간 + DB 접근시간)**



Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지

Hit Rate 80%

Miss Rate 20%

Cache 접근시간 0.1 μ s

DB 접근시간 100 μ s

Hit Rate x Cache 접근시간

+ **Miss Rate x (Cache 접근시간 + DB 접근시간)**

$$= 0.8 \times 0.1\mu\text{s} + 0.2 \times (0.1\mu\text{s} + 100\mu\text{s})$$

$$= 20.1\mu\text{s}$$

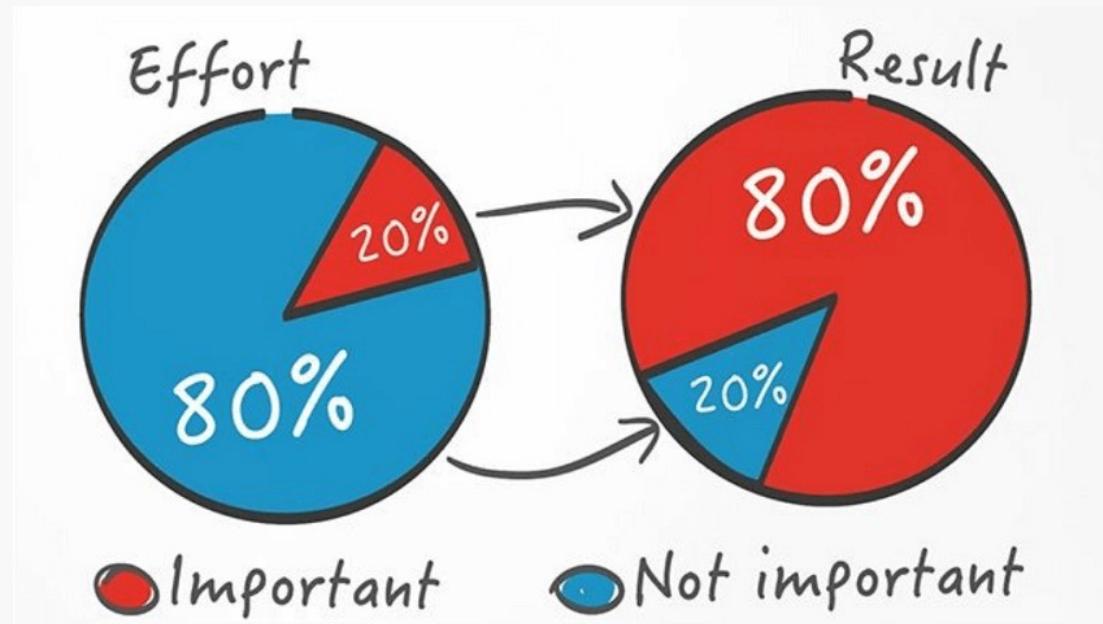
Cache



왜 사용할까?

파레토 법칙

- 중요한 20%가 전체의 80%를 차지



Cache



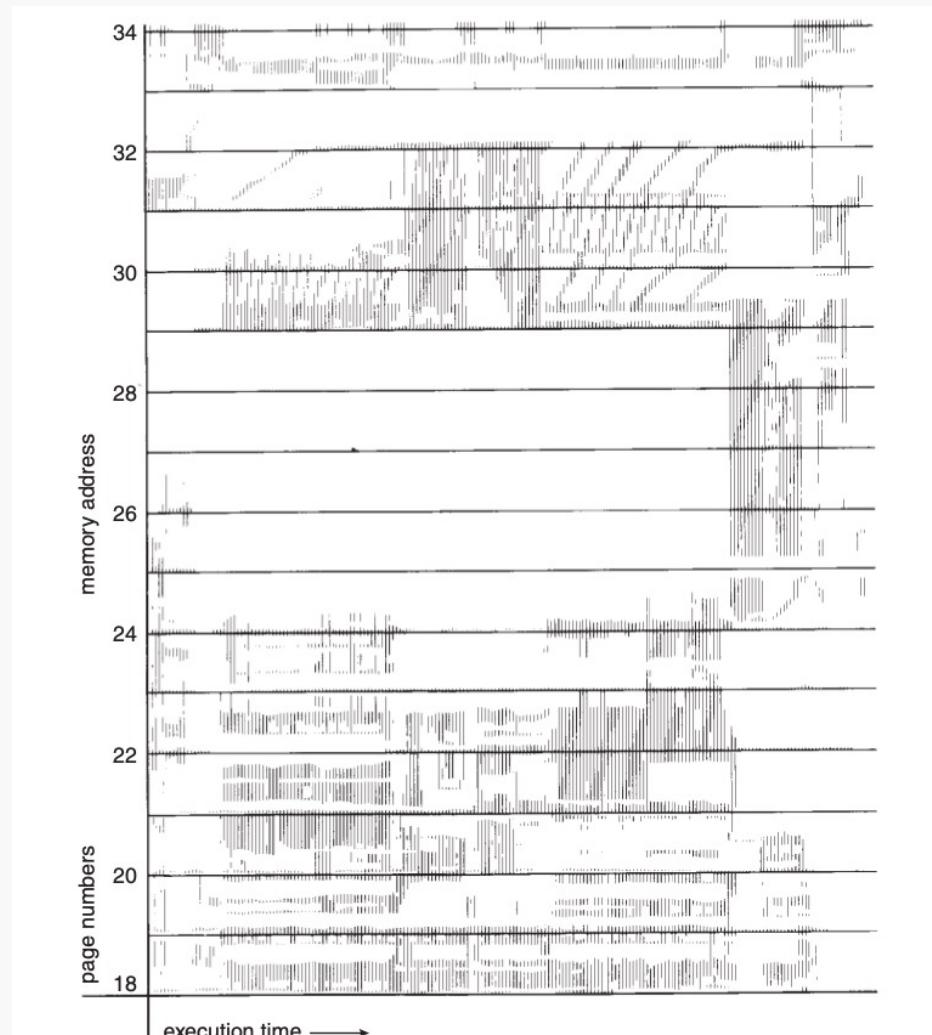
참조의 지역성

동일한 값이나 관련 범위의 데이터가 자주 접근되는 특성

시간 지역성

공간 지역성

순차 지역성



출처: <https://woodforest.tistory.com/15>

Cache



왜 사용할까?

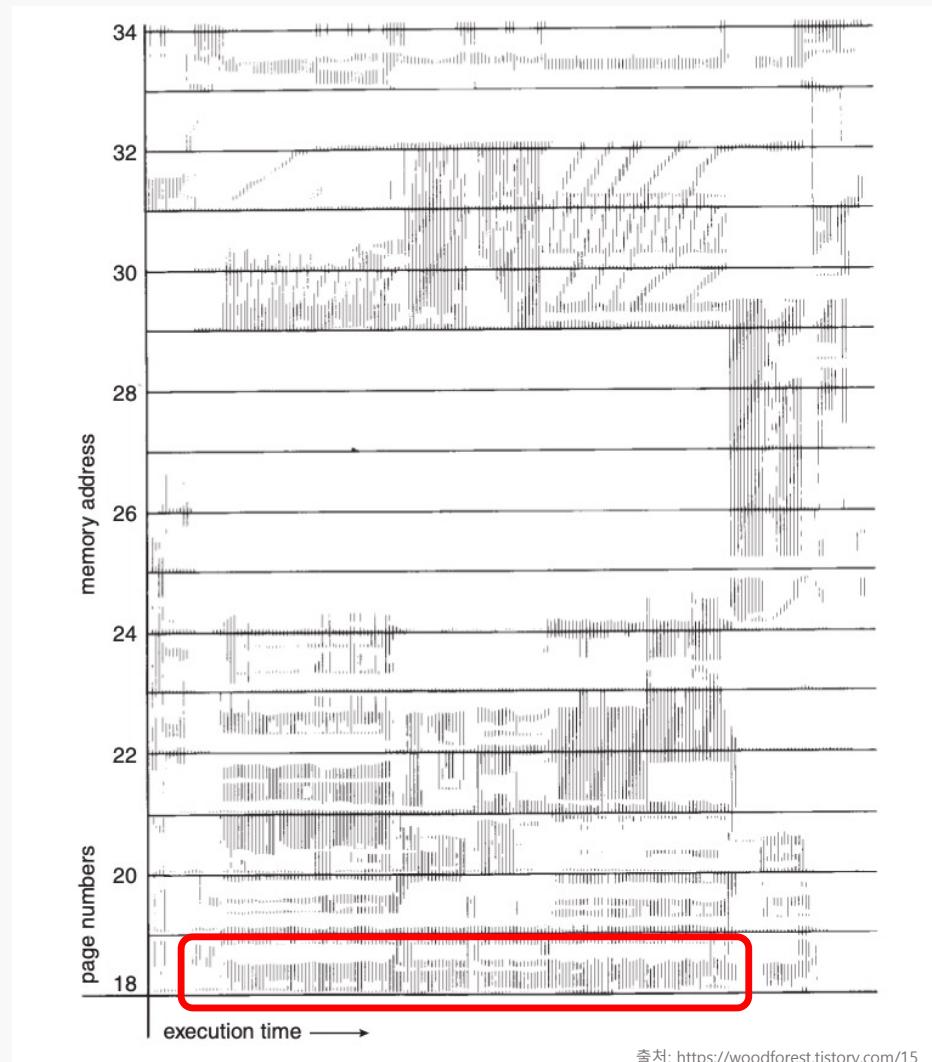
참조의 지역성

동일한 값이나 관련 범위의 데이터가 자주 접근되는 특성

시간 지역성

공간 지역성

순차 지역성



출처: <https://woodforest.tistory.com/15>

Cache



왜 사용할까?

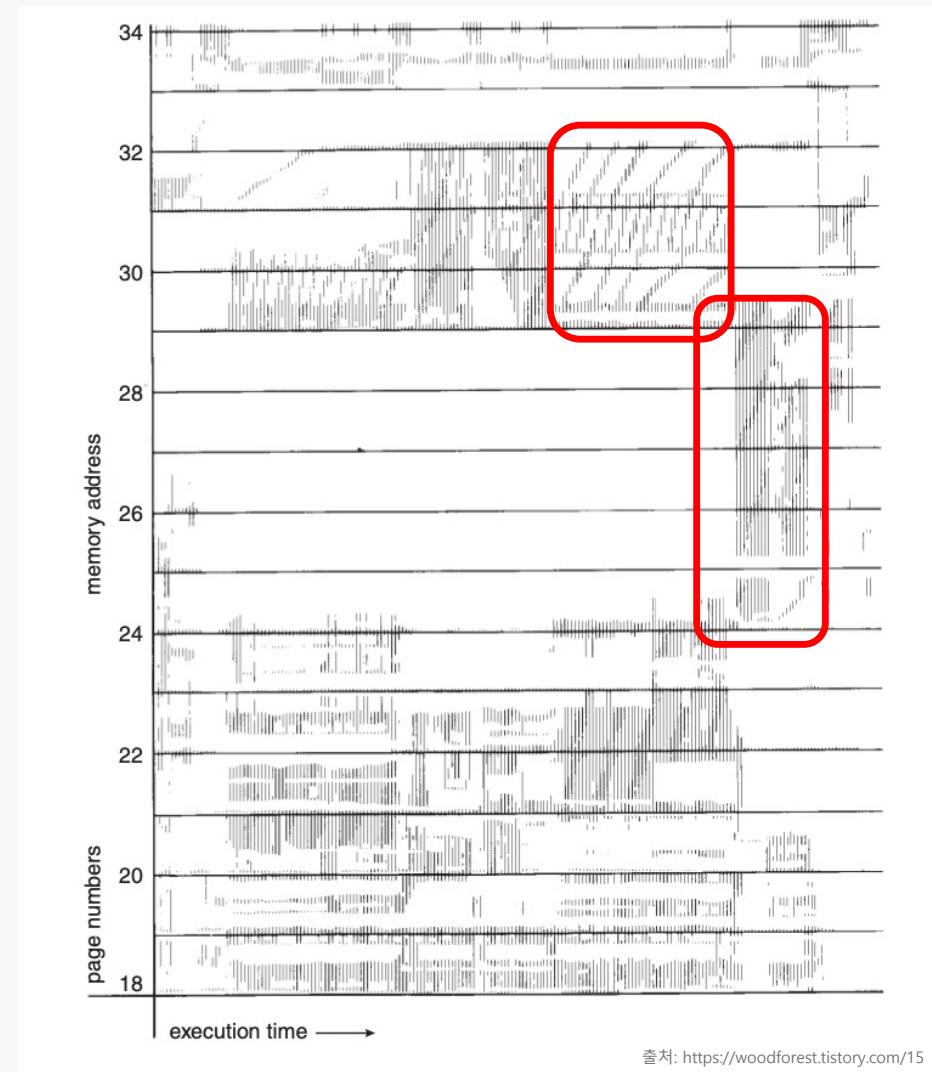
참조의 지역성

동일한 값이나 관련 범위의 데이터가 자주 접근되는 특성

시간 지역성

공간 지역성

순차 지역성



Cache



캐시 쓰기 전략

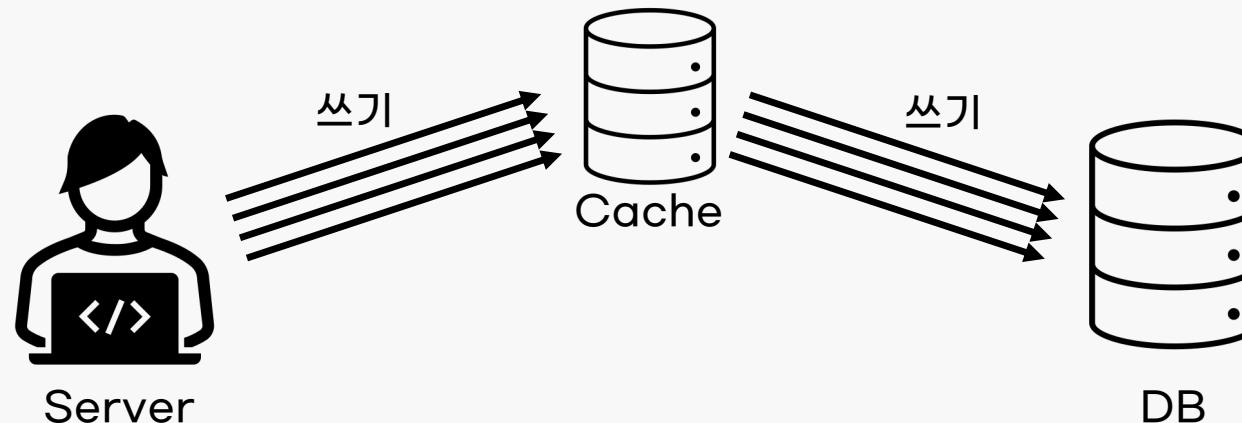
Write Through

장점

데이터 정합성 보장

단점

항상 두번의 쓰기 연산 발생



Cache

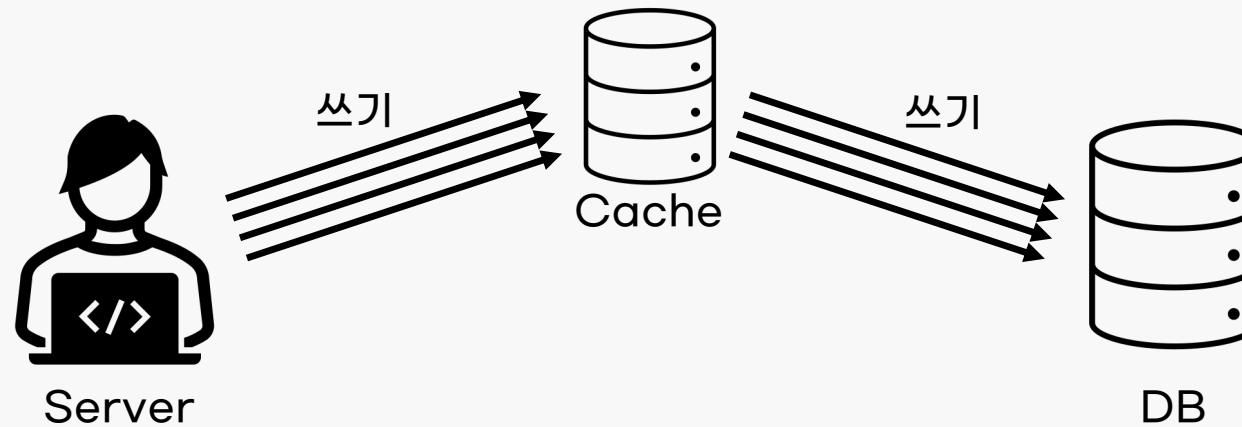


캐시 쓰기 전략

Write Through

언제 사용할까?

데이터 유실이 발생하면 안 되는 경우
쓰기보다 읽기 요청이 압도적인 경우



Cache



캐시 쓰기 전략

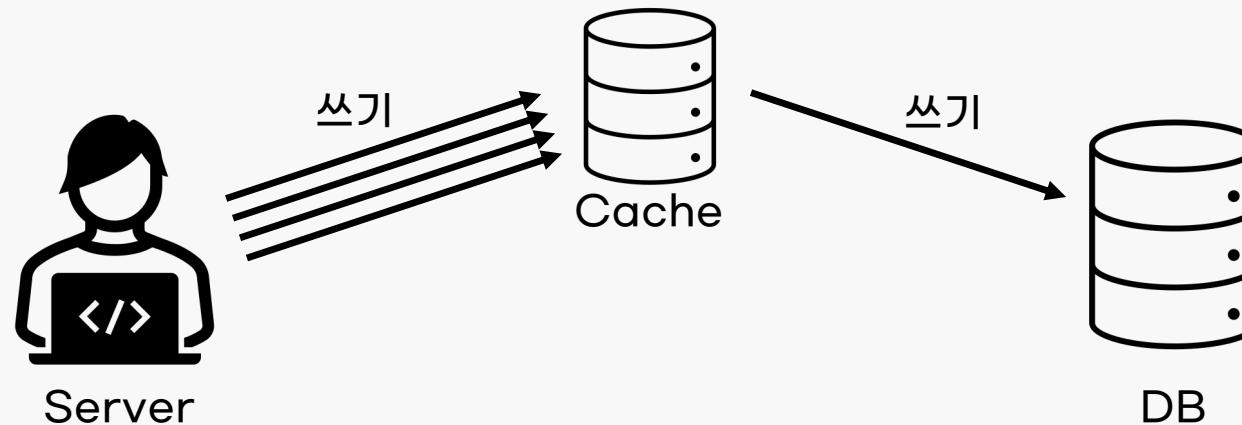
Write Back

장점

쓰기 과정의 오버헤드 감소

단점

캐시 데이터 유실 가능성 존재



Cache

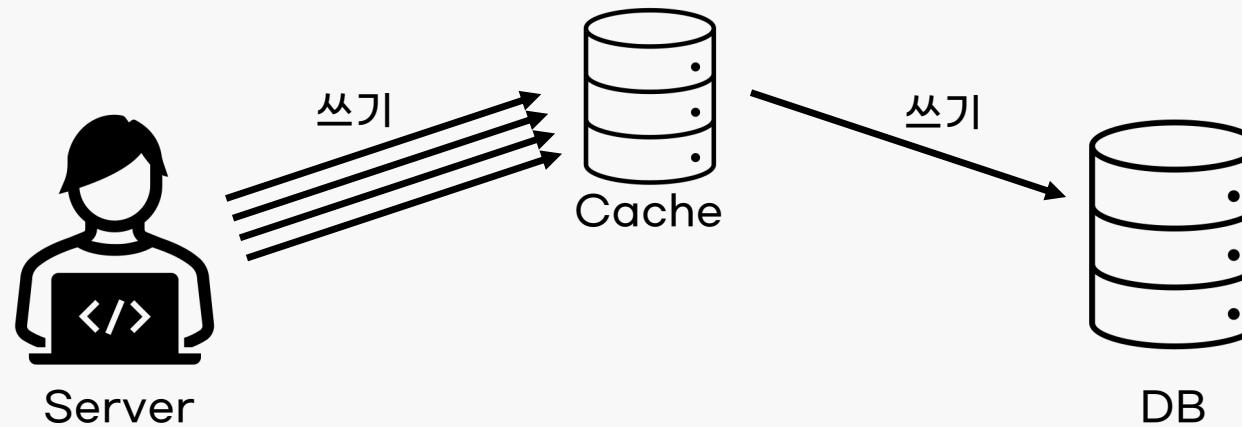


캐시 쓰기 전략

Write Back

언제 사용할까?

어느정도 데이터 유실이 상관없는 경우
쓰기 연산이 자주 호출되는 경우



Content

Cache

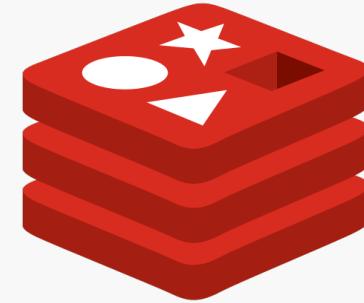
Redis

ElastiCache



Redis란?

**Remote
Dictionary
Server**

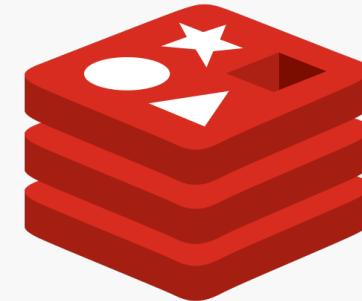




Redis란?

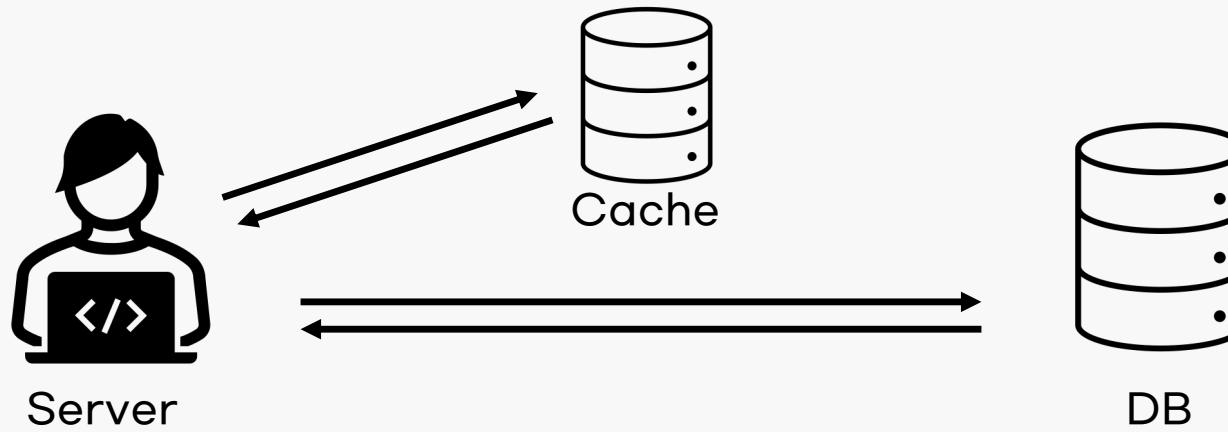
**Remote
Dictionary
Server**

원격
Key-Value
서버



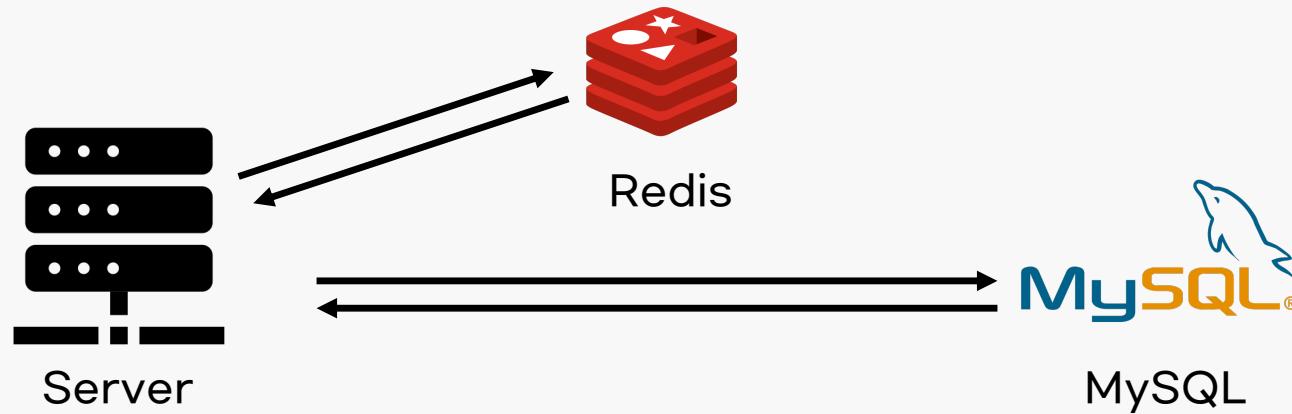
Redis

Redis란?



Redis

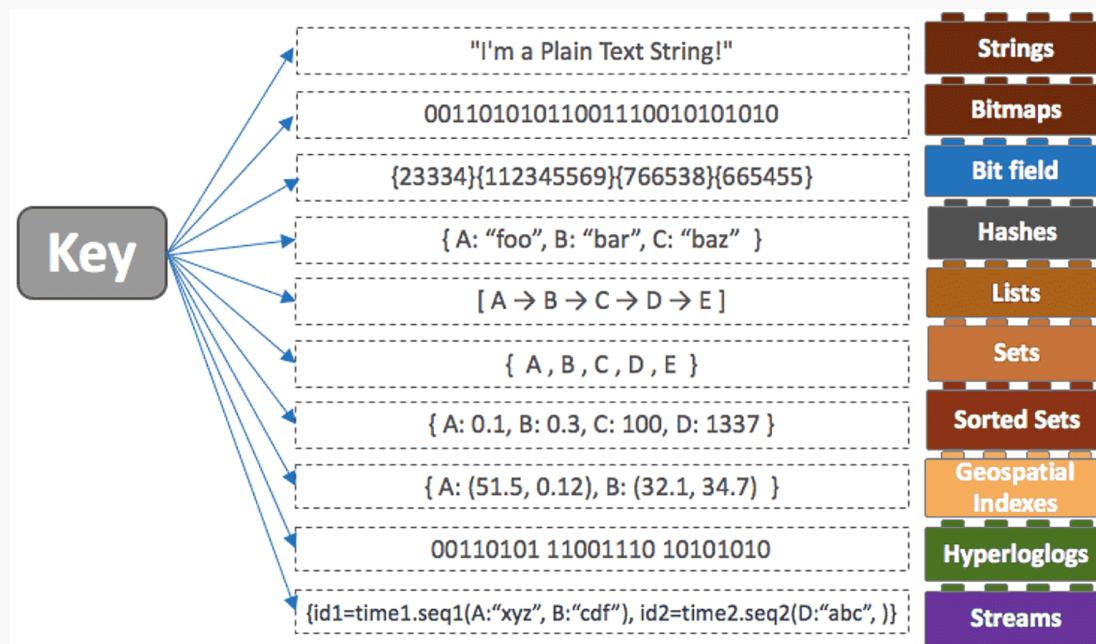
Redis란?





Redis 특징

다양한 Collections



출처: <https://sjh836.tistory.com/178>



Redis 특징

Key-Value 형태

전화번호부

Key	Value
김철수	010-1111-1111
박영희	010-2222-2222
이동주	010-3333-3333

데이터 쓰기

```
set {key} {value}  
set 김철수 010-1111-1111
```

데이터 읽기

```
get {key}  
get 김철수
```



Redis 특징

Key-Value 형태

전화번호부

Key	Value
김철수	010-1111-1111
박영희	010-2222-2222
이동주	010-3333-3333

데이터 쓰기

```
set {key} {value}  
set 김철수 010-1111-1111
```

데이터 읽기

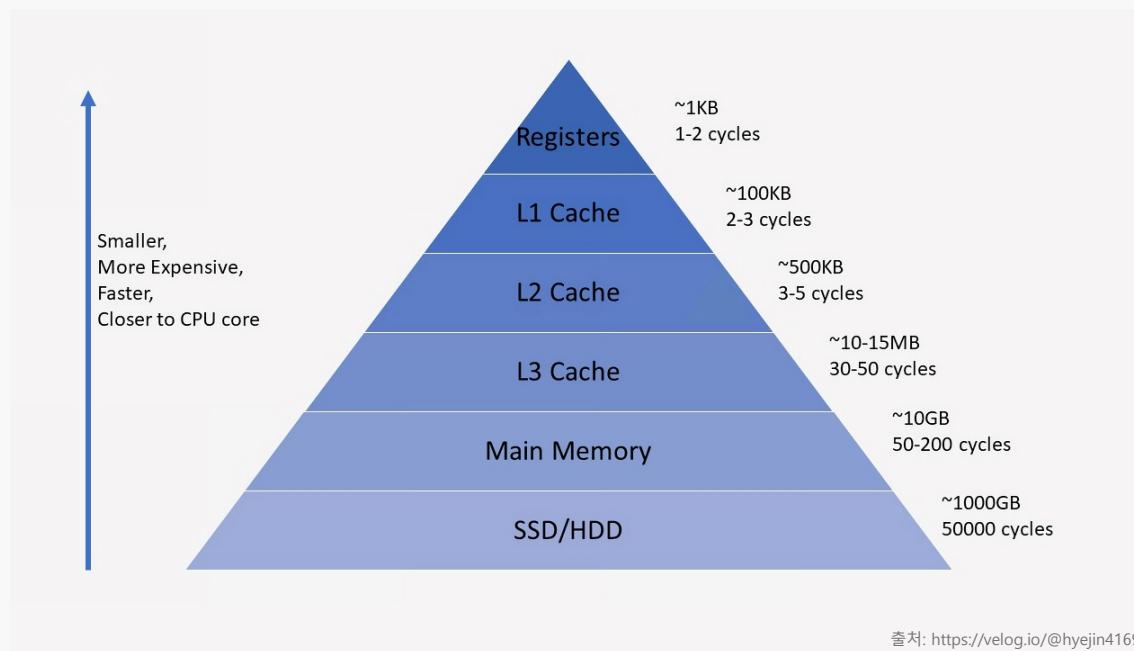
```
get {key}  
get 김철수
```

"이론적으로 초당 10만 TPS 이상 가능"



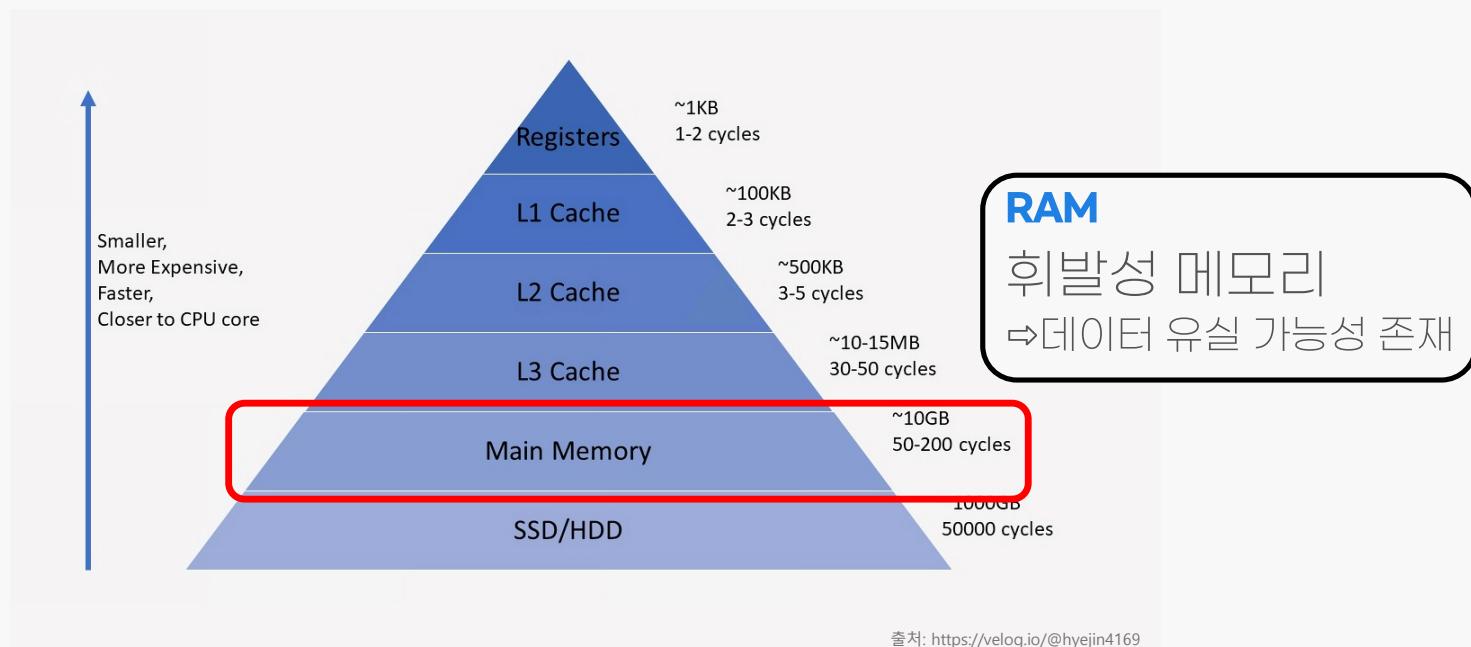
Redis 특징

In-Memory Store



Redis 특징

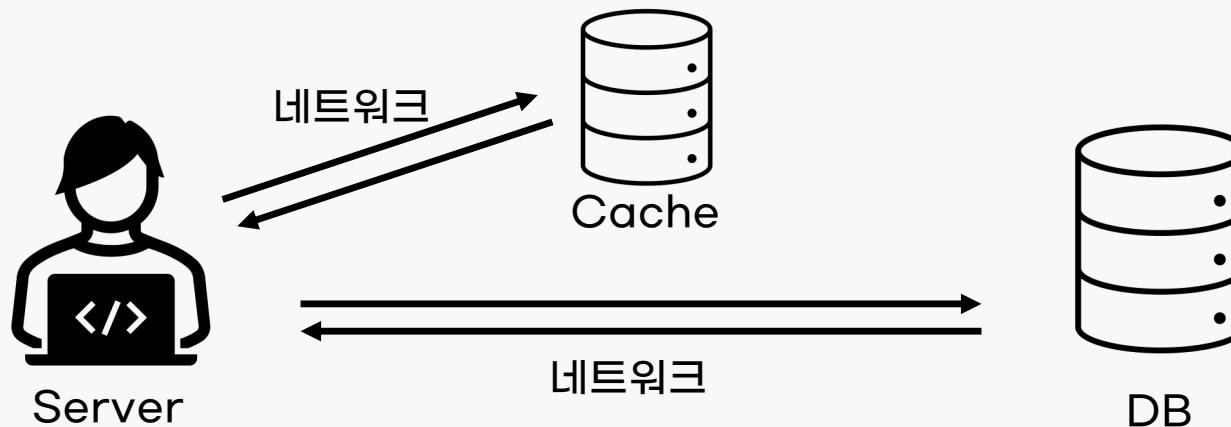
In-Memory Store





Redis 특징

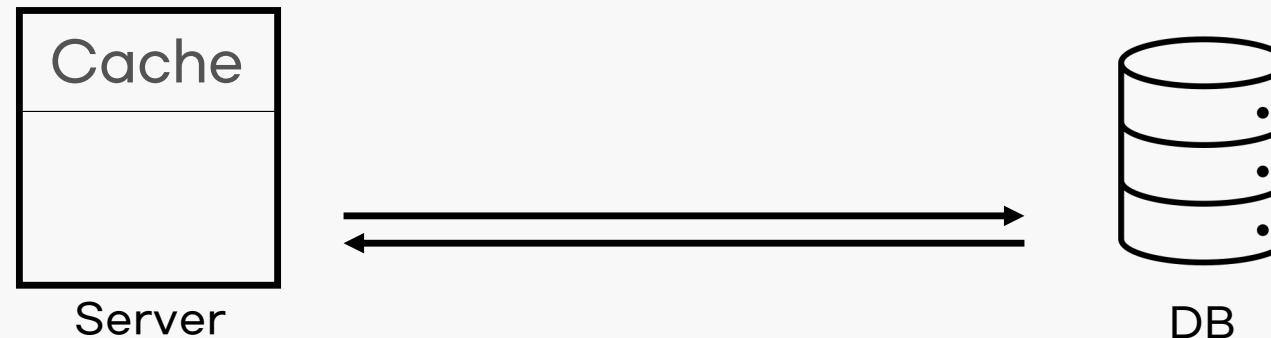
Global 캐시





Redis 특징

Global 캐시 *Local 캐시

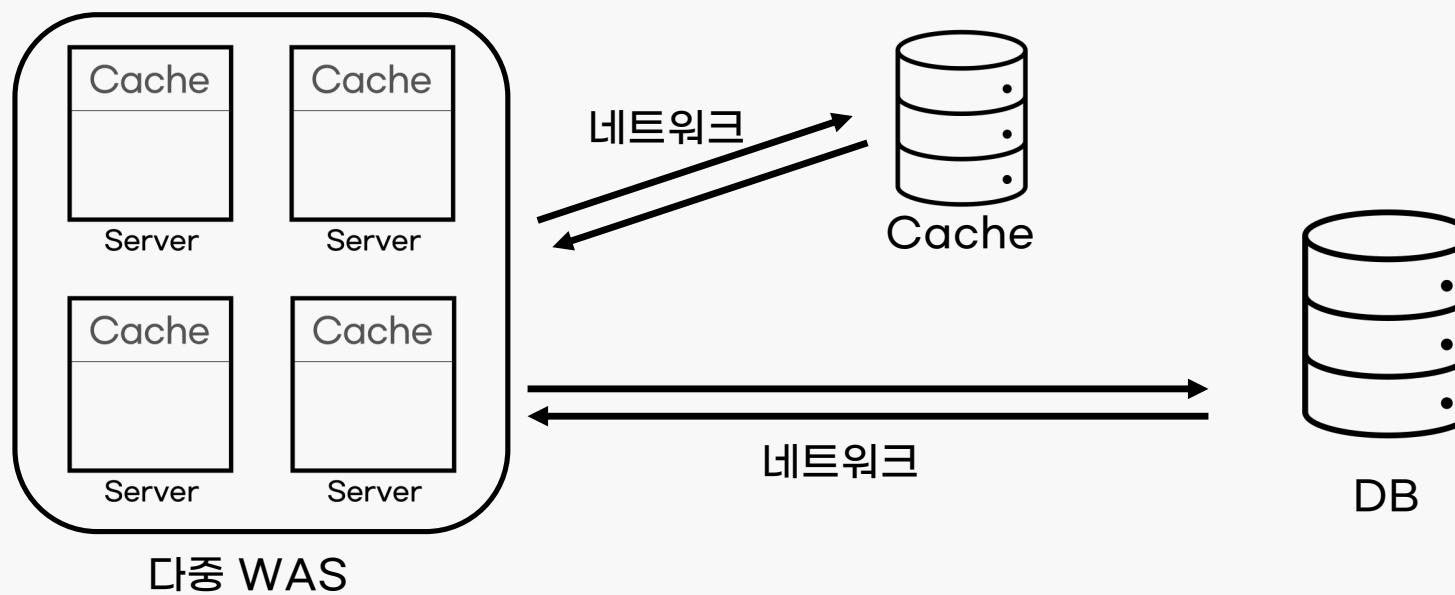


Local Cache의 대략적 구조



Redis 특징

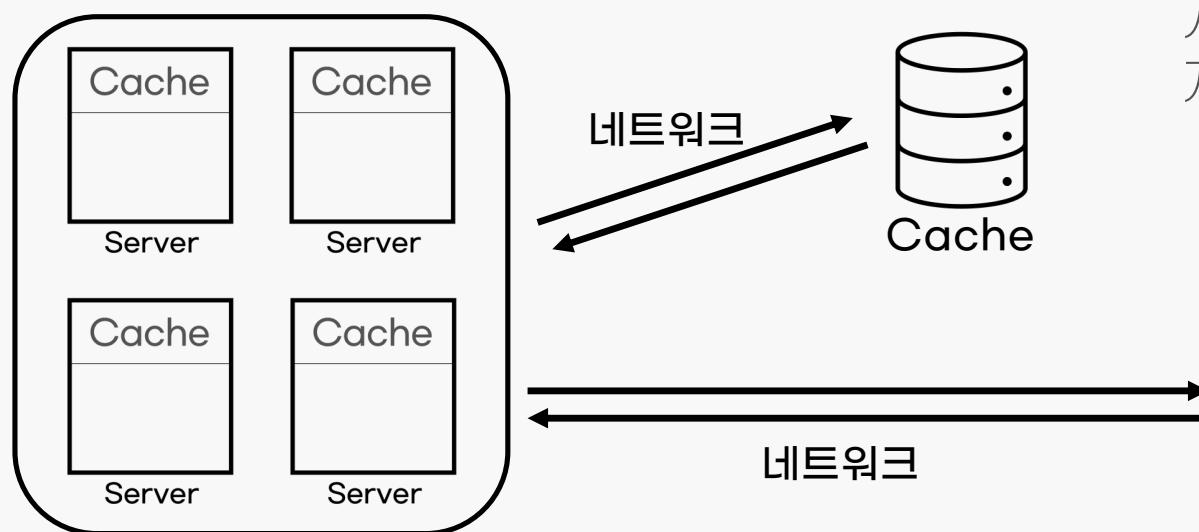
Global 캐시





Redis 특징

Global 캐시



Local 캐시

네트워크 오버헤드가 없음

Global 캐시

서버 간 데이터 공유가 쉬움
자원을 별도로 사용 가능

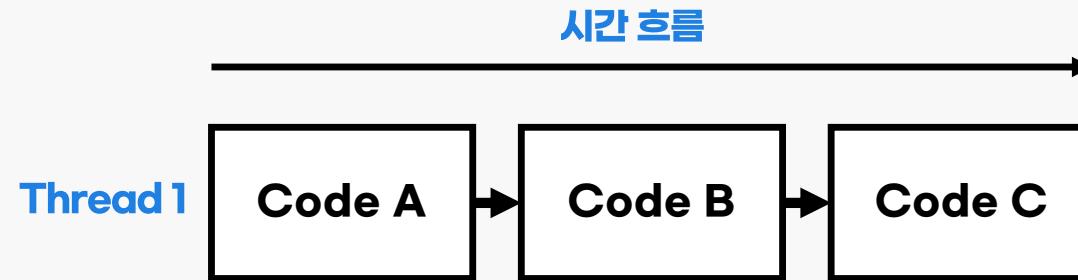




Redis 특징

Single Thread

"Single
Thread "

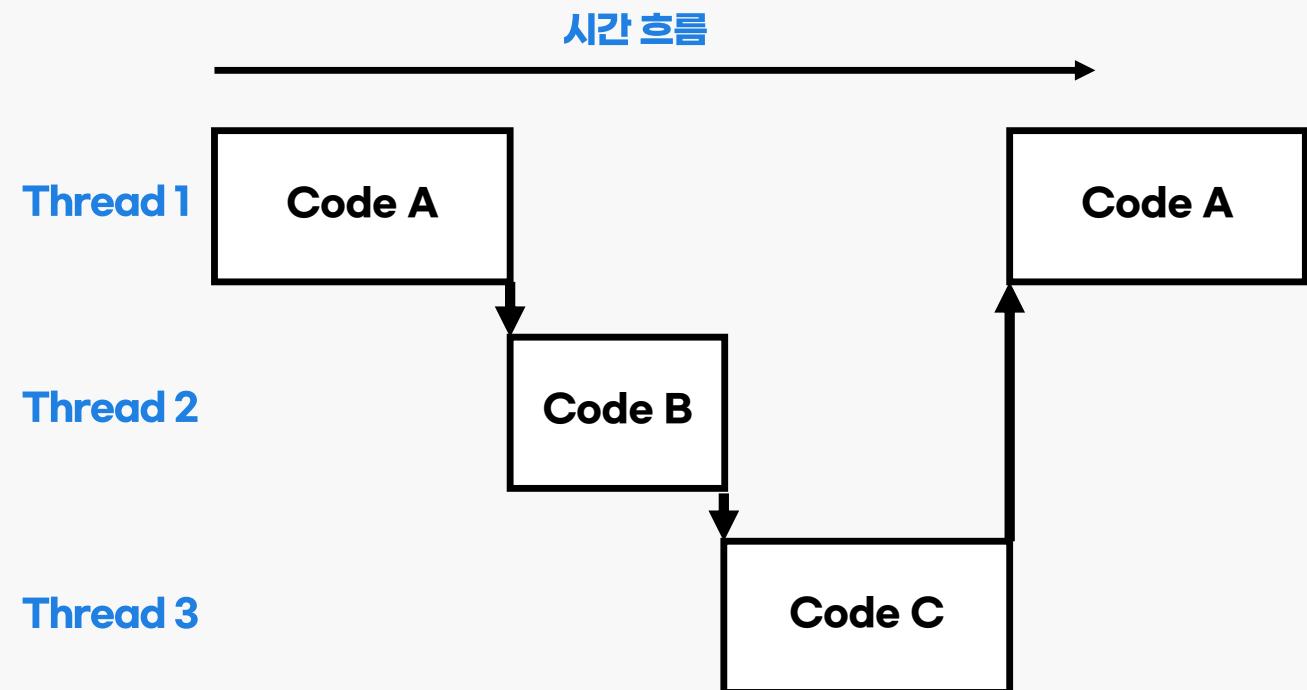




Redis 특징

Single Thread

"Multi
Thread "





Redis 특징

Single Thread

Single Thread

- 단순하게 구현 가능
- 효율적인 메모리 사용
- 동기화 문제 발생 X (Atomic)
- 한 작업의 수행시간이 길다면
나머지 작업의 대기시간이 길어짐

Multi Thread

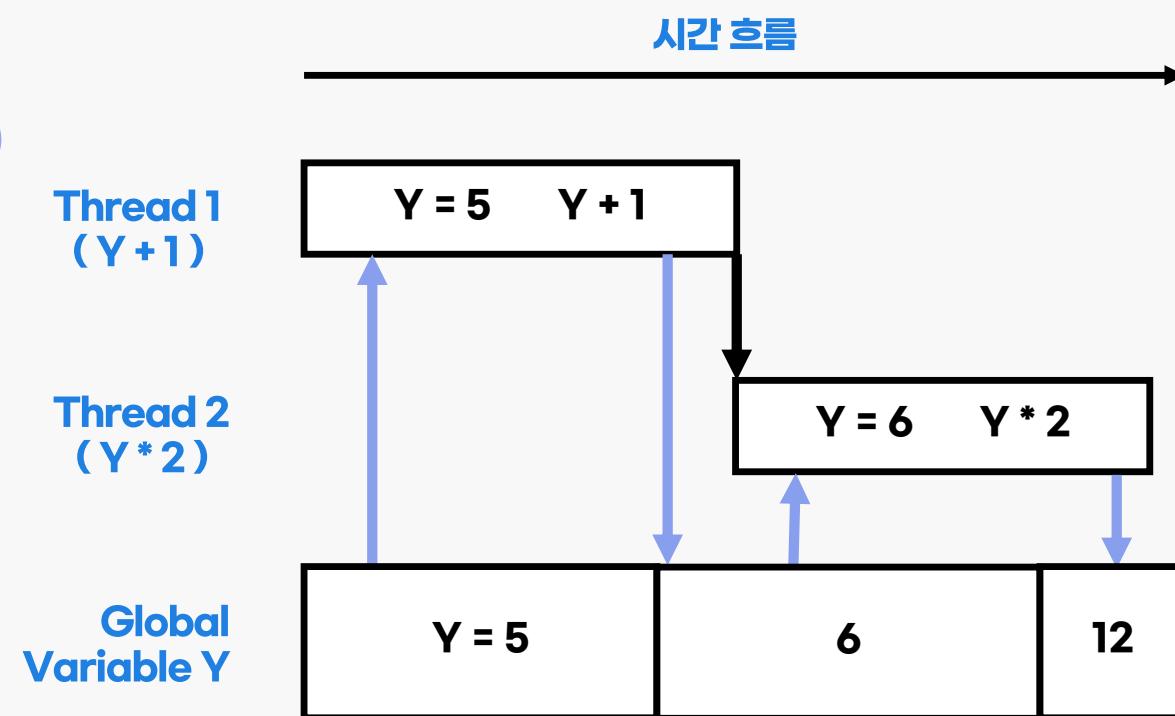
- 구현 방식이 복잡함
- 스레드 교환에 메모리 오버헤드 발생
- 동기화 문제를 해결하기 위한 추가 코드가 필요함
- 한 작업의 수행시간이 길어도
빠른 응답 속도를 볼 수 있음



Redis 특징

Multi Thread의 문제점

"Race
Condition"

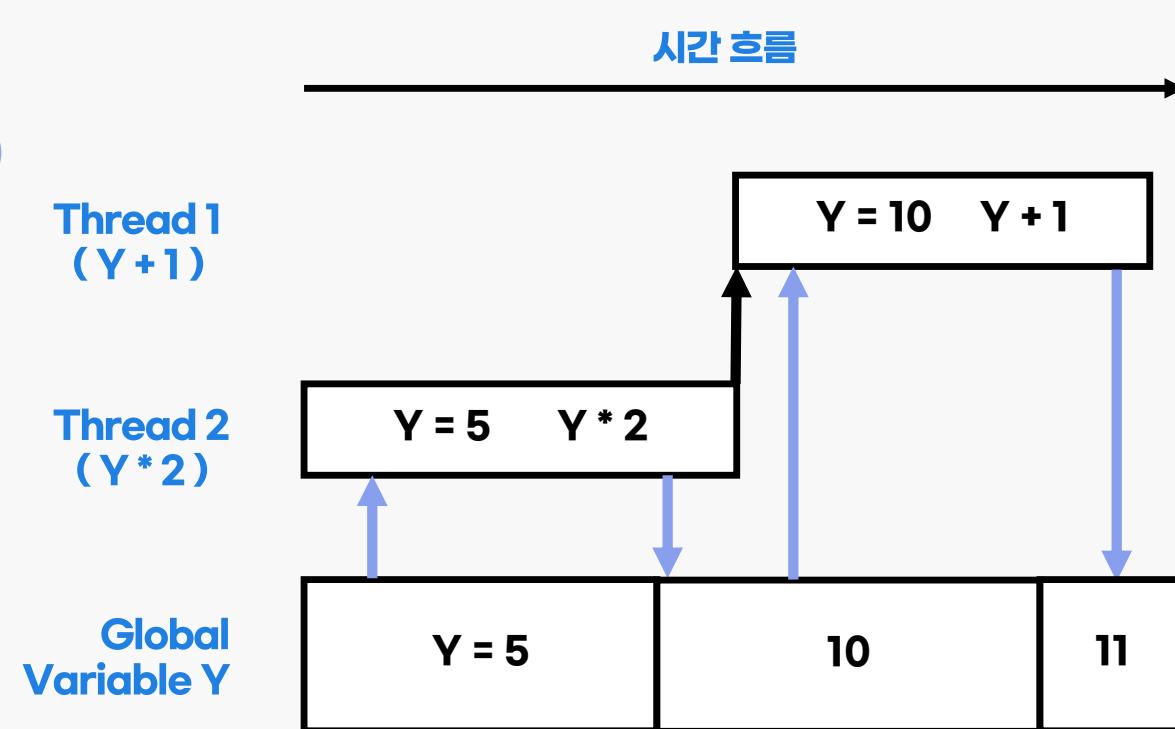




Redis 특징

Multi Thread의 문제점

"Race
Condition"

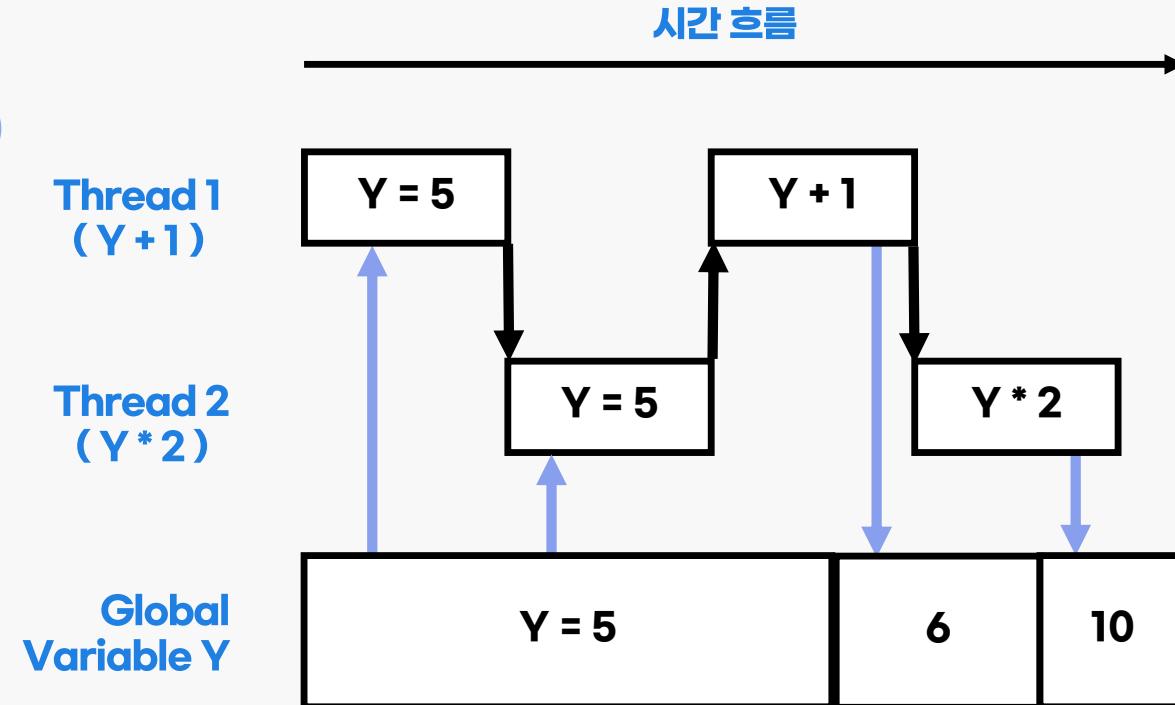




Redis 특징

Multi Thread의 문제점

"Race Condition"

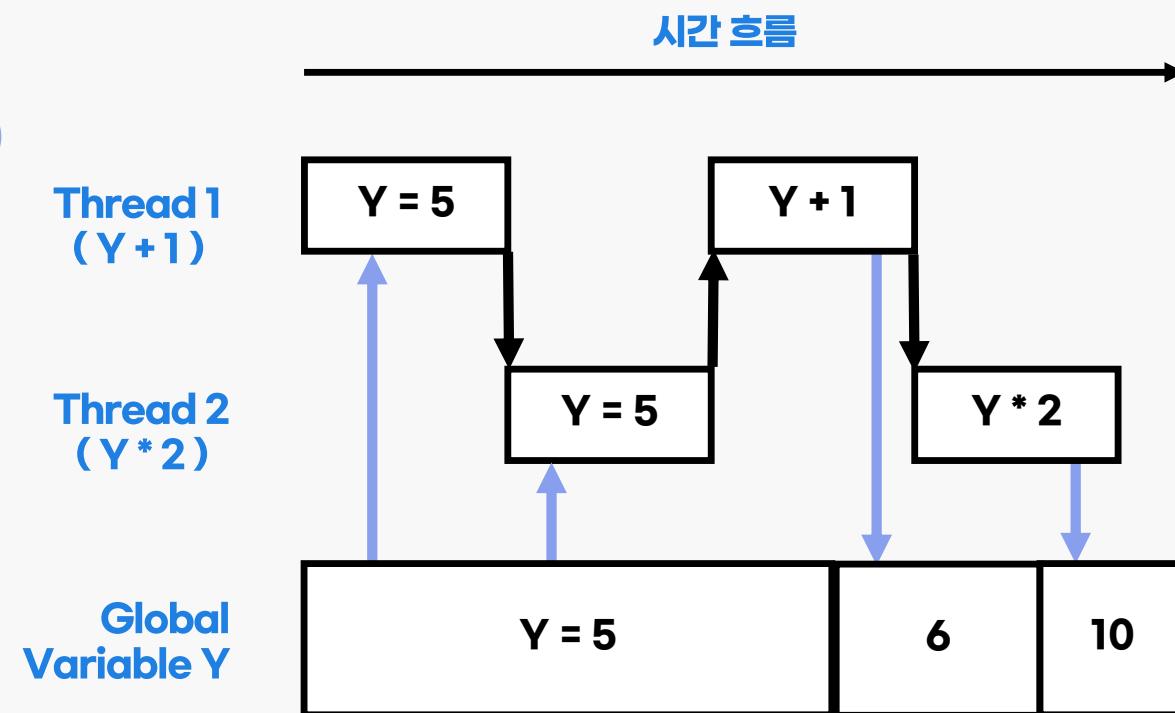




Redis 특징

Single Thread

"Race
Condition"



Multi Thread에서의 문제 - 동기화 작업 필요



Redis 특징

사실 쓰레드는 여러개다

명령 실행은 Single Thread로 처리
+ 보조 Thread로 독립된 기능 처리



Redis 특징

사실 쓰레드는 여러개다

Redis Github bio.h

```
38     /* Background job opcodes */
39     #define BIO_CLOSE_FILE      0 /* Deferred close(2) syscall. */
40     #define BIO_AOF_FSYNC        1 /* Deferred AOF fsync. */
41     #define BIO_LAZY_FREE        2 /* Deferred objects freeing. */
42     #define BIO_NUM_OPS          3
```

Main Thread Redis를 실행했을 때 동작하는 쓰레드

Sub Thread 1 BIO_CLOSE_FILE - AOF Rewrite시 사용하는 쓰레드

Sub Thread 2 BIO_AOF_FSYNC - AOF 동기화 쓰레드

Sub Thread 3 BIO_LAZY_FREE - UNLINK, 비동기



Redis 특징

사실 쓰레드는 여러개다

네트워크 부분에서의 R/W를 Multi Thread로 처리

By delegating the time spent reading and writing to I/O sockets over to other threads, the Redis process can devote more cycles to manipulating, storing, and retrieving data—boosting overall performance. This improvement retains the transactional characteristics of previous versions, so you don't have to rethink your applications to take advantage of the increased performance. Similarly, Redis' single-threaded **DEL** command can now be configured to behave like the multi-thread **UNLINK** command that has been available since Redis version 4.

출처: <https://redis.io/blog/diving-into-redis-6/>



Redis 특징

사실 쓰레드는 여러개다

"Multi Thread로 구성이 되어있지만
명령 실행은 Single Thread로 처리"

⇒ Atomic 보장

Content

Cache

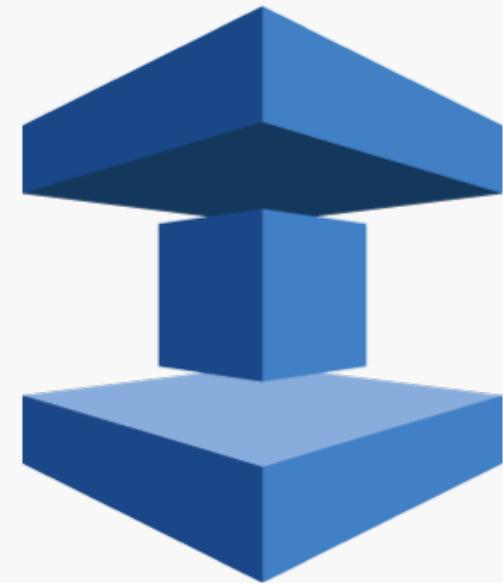
Redis

ElastiCache



ElastiCache란?

클라우드에서
인메모리 캐시를
설정, 운영, 확장 가능한
관리형 서비스



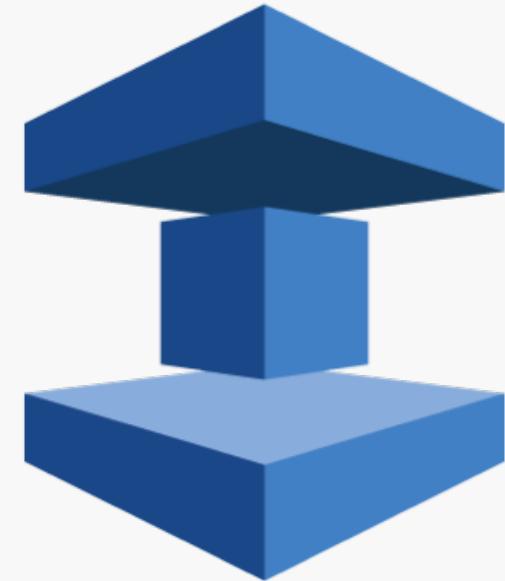


ElastiCache란?

Redis 서버 생성

- + 관리 자동화
- + 쉬운 설정, 배포
- + 멀티 AZ 지원, 자동백업
- + 수평, 수직 확장

...





ElastiCache

The screenshot shows the Amazon ElastiCache landing page on the AWS website. The top navigation bar includes the AWS logo, a search bar, and various navigation icons. The main heading is "Amazon ElastiCache" with the subtitle "실시간 애플리케이션을 위한 실시간 성능". A descriptive text below states: "실시간 성능을 제공하고 애플리케이션을 즉시 확장할 수 있도록 합니다. ElastiCache는 두 개의 오픈 소스 캐싱 솔루션인 Redis 및 Memcached와 호환됩니다." A prominent orange button labeled "지금 시작" (Start Now) is visible, along with links for "Redis" and "Memcached". The footer contains standard AWS links: CloudShell, 의견, 개인 정보 보호, 약관, and 쿠키 기본 설정, along with the copyright notice: © 2024, Amazon Web Services, Inc. 또는 계열사.



ElastiCache

The screenshot shows the AWS ElastiCache console interface for creating a Redis cache cluster. The top navigation bar includes the AWS logo, service menu, search bar, and Seoul region selection. The main title is "ElastiCache > ... > Redis 캐시 생성".

생성 방법 (Creation Method):

- 서비스 - 신규**: Serverless management without application tracing, automatically scaling the cache to handle peak loads.
- 자체 캐시 설계**: Custom design for node type, size, and count.
- 간편한 생성**: Quick creation using cluster template options.
- 클러스터 캐시**: Restore from a backup or RDB file.
- 백업에서 복원**: Restore from a backup or RDB file.

클러스터 모드 (Cluster Mode):

자동 중단 없이 클러스터 크기를 동적으로 조정합니다.

- 활성화됨**: Clusters have auto-scaling and failover.
- 비활성화됨**: Redis clusters have a primary node and up to 5 secondary replicas per shard.

CloudShell 의견 © 2024, Amazon Web Services, Inc. 또는 계열사.



ElastiCache 클러스터

클러스터 모드 활성화

클러스터 모드 비활성화

The screenshot shows the AWS ElastiCache Redis Cache Creation interface. At the top, there are two main options for cache creation:

- 서비스 - 신규: Manages servers directly without application-level replication.
- 자체 캐시 설계: Allows users to select node types, sizes, and counts to build their own cache.

Below these are three methods for cache creation:

- 간편한 생성: Creates a cluster using standard configurations.
- 클러스터 캐시: Creates a cluster using all available configuration options.
- 백업에서 복원: Restores a cluster from a backup or RDB file.

A section titled "클러스터 모드" (Cluster Mode) allows for dynamic scaling of cluster size. It contains two options:

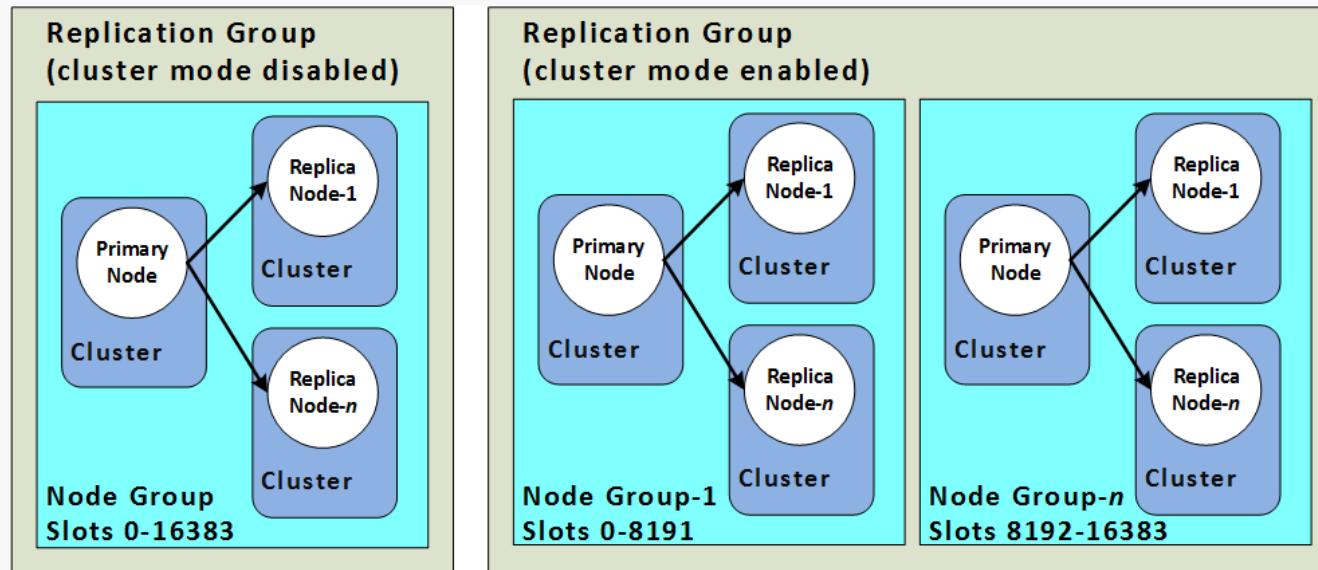
- 활성화됨: Clusters can be scaled horizontally and vertically.
- 비활성화됨: Redis clusters have a single primary node and up to 5 secondary nodes.

At the bottom, there are footer links for CloudShell,的意见 (Feedback), 개인정보 보호 (Privacy), 약관 (Terms), and 쿠키 기본 설정 (Cookie Settings). A copyright notice at the bottom right states: © 2024, Amazon Web Services, Inc. 또는 계열사.

ElastiCache



ElastiCache 클러스터



출처: <https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Replication.Redis-RedisCluster.html>



ElastiCache 클러스터

클러스터 모드 비활성화

Node Group





ElastiCache 클러스터

클러스터 모드 비활성화

Node Group

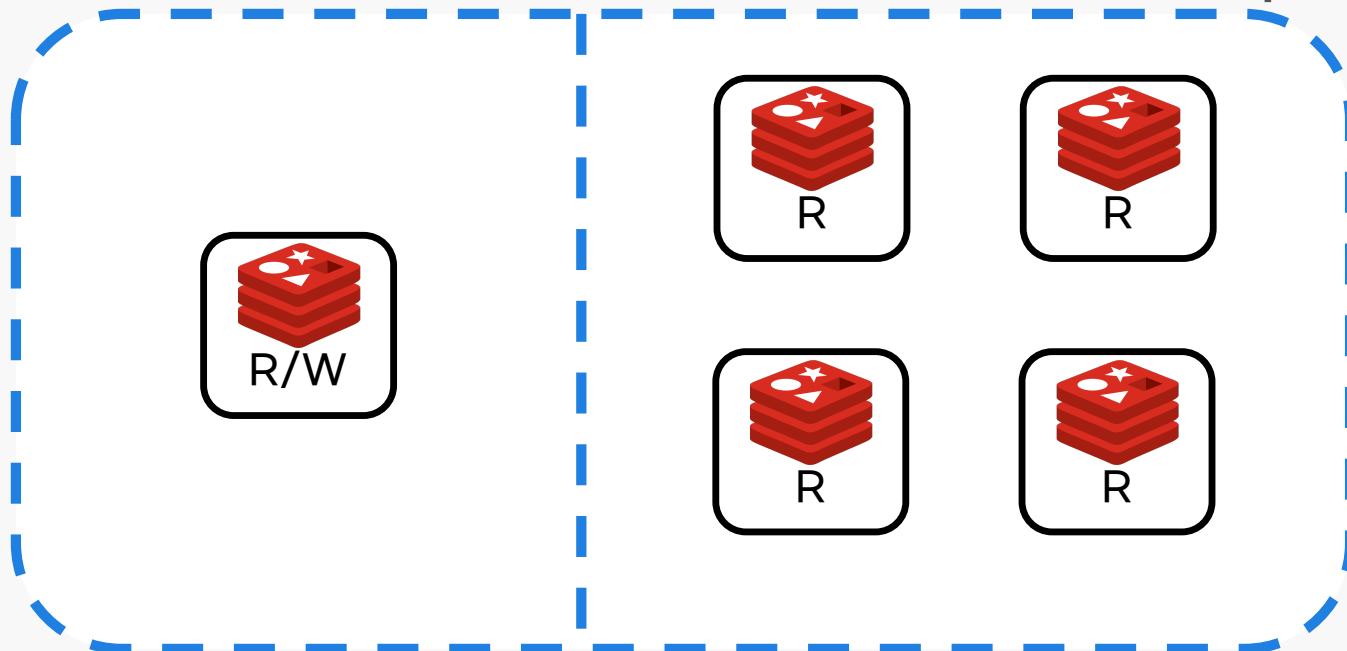




ElastiCache 클러스터

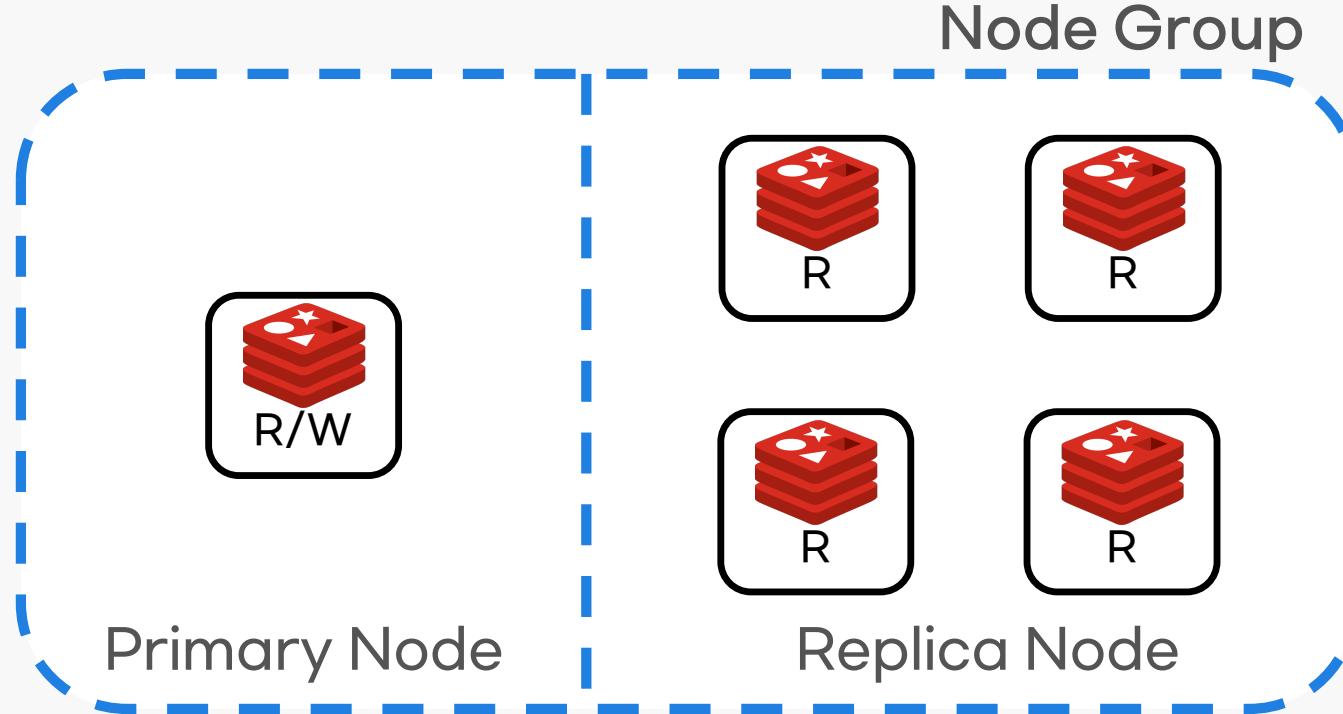
클러스터 모드 비활성화

Node Group





ElastiCache 클러스터 클러스터 모드 비활성화





ElastiCache 클러스터

클러스터 모드 활성화

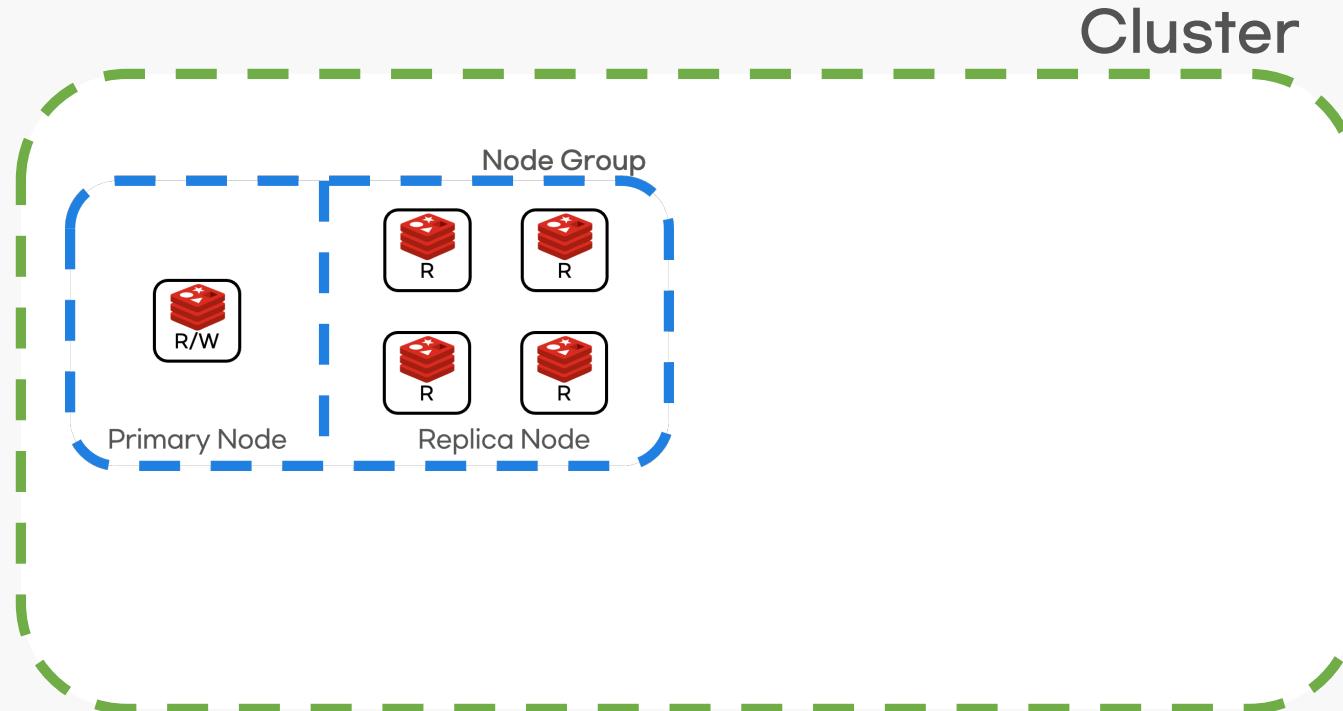
Cluster





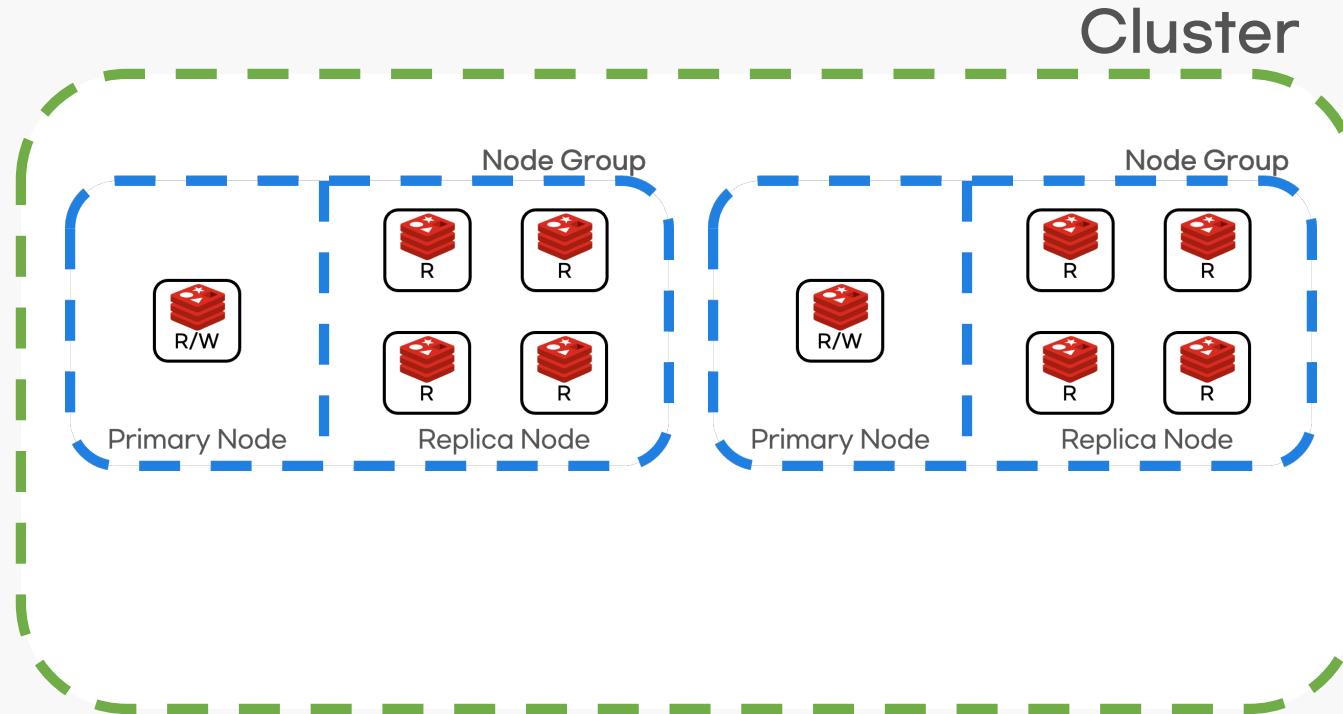
ElastiCache 클러스터

클러스터 모드 활성화



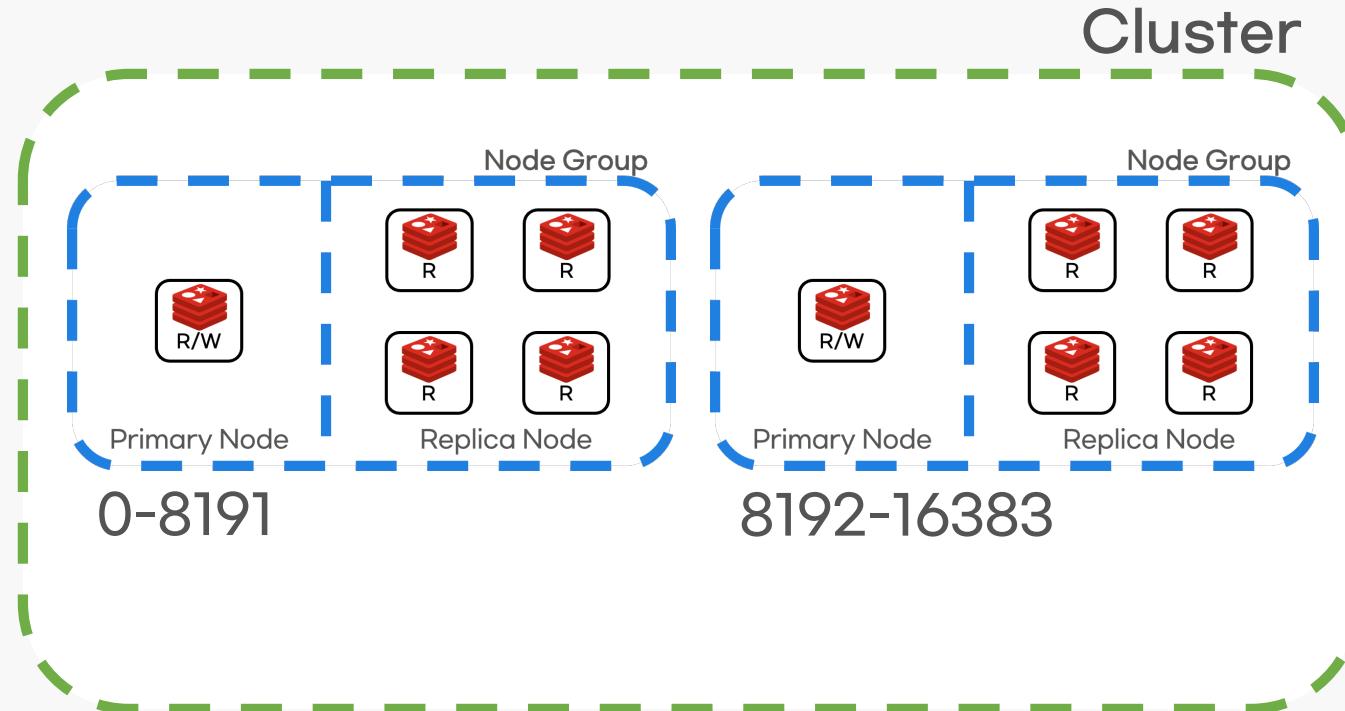


ElastiCache 클러스터 클러스터 모드 활성화





ElastiCache 클러스터 클러스터 모드 활성화





ElastiCache 클러스터

	비활성화	활성화
Read Replicas	0 ~ 5개	
Shards	1개	1 ~ 500개
Partitioning	불가능	가능
Scaling	Vertical	Vertical Horizontal
Multi-AZ	Node Group별 Replica 수가 1개 이상이면 사용 가능	



ElastiCache

aws | 서비스 | 서울 | 1/3 단계 | ElastiCache > ... > Redis 캐시 생성 | ⓘ | ⓘ

클러스터 설정 정보

구성 정보
다음 옵션 중 하나를 선택하여 Redis 캐시를 생성합니다.

배포 옵션

- 서비스 - 신규
서버를 관리하지 않고도 애플리케이션 트래픽 수요에 맞게 자동으로 확장되는 캐시를 빠르게 생성하는 데 사용합니다.
- 자체 캐시 설계
노드 유형, 크기, 개수를 선택하여 캐시를 생성하는 데 사용합니다.

생성 방법

- 간편한 생성
권장 모범 사례 구성을 사용합니다. 클러스터를 생성한 후 옵션을 수정할 수도 있습니다.
- 클러스터 캐시
새 클러스터의 모든 구성 옵션 설정
- 백업에서 복원
기존 백업 또는.rdb 파일을 사용하여 클러스터를 복원합니다.

클러스터 모드

자동 중단 없이 클러스터 크기를 동적으로 조정합니다.

- 활성화됨
클러스터 모드에서는 확장성과 가용성을 개선 하도록 여러 샤드에서 복제할 수 있습니다.
- 비활성화됨
Redis 클러스터에는 프라이머리 노드 하나와 최대 5개의 읽기 전용 복제본을 포함하는 단일 샤드(노드 그룹)가 있습니다.

① 클러스터 모드를 사용하면 최대 500개의 노드 그룹에서 데이터 파티셔닝을 지원하고 Redis 클러스터의 성능을 개선할 수 있습니다. 이 모드에서는 일부 명령을 사용할 수 없습니다. [자세히 알아보기](#)

클러스터 정보



ElastiCache

샤드(노드 그룹)가 있습니다.

클러스터 모드를 사용하면 최대 500개의 노드 그룹에서 데이터 파티셔닝을 지원하고 Redis 클러스터의 성능을 개선할 수 있습니다. 이 모드에서는 일부 명령을 사용할 수 없습니다. [자세히 알아보기](#)

클러스터 정보

다음 옵션을 사용하여 클러스터를 구성합니다.

이름

qwer

이름은 최대 40자까지 가능하며 공백을 포함할 수 없습니다.

설명 - 선택 사항

설명

위치

클러스터를 AWS 클라우드에서 호스팅할지, 온프레미스에서 호스팅할지 선택합니다.

위치

AWS 클라우드

ElastiCache 인스턴스에 대해 AWS 클라우드를 사용합니다.

온프레미스

AWS Outposts를 통해 Outposts에서 ElastiCache 인스턴스를 생성합니다. 먼저 Outpost에서 서브넷 ID를 생성해야 합니다.

다중 AZ

사용

다중 AZ는 프라이머리 노드에서 장애가 발생한 경우 여러 AZ에서 읽기 전용 복제본으로 자동 장애 조치를 수행하여 향상된 고가용성을 제공합니다.

자동 장애 조치

사용

ElastiCache 자동 장애 조치는 프라이머리 노드에서 장애 조치가 발생할 경우 읽기 전용 복제본으로 자동 장애 조치를 통해 향상된 고가용성을 제공합니다.

클러스터 설정

다음 옵션을 사용하여 클러스터를 구성합니다.



ElastiCache

시중 경매 소식

사용
ElastiCache 자동 장애 조치는 프라이머리 노드에서 장애 조치가 발생할 경우 일기 전용 복제본으로 자동 장애 조치를 통해 향상된 고가용성을 제공합니다.

클러스터 설정

다음 옵션을 사용하여 클러스터를 구성합니다.

엔진 버전

해당 노드에서 실행될 Redis 엔진의 버전 호환성

7.1

포트

노드가 연결을 허용하는 포트 번호입니다.

6379

파라미터 그룹

파라미터 그룹은 노드 및 클러스터의 런타임 속성을 제어합니다.

default.redis7.cluster.on



노드 유형

배포할 노드 유형 및 연결된 메모리 크기입니다.

cache.r6g.large

13.07 GiB memory Up to 10 Gigabit network performance

shard 수

이 클러스터의 shard 수 입력(1~500개)

3

shard당 복제본 수

각 shard에 대한 복제본 수(0~5개)를 입력합니다.

1

연결

이 클러스터가 지원할 IP 버전을 선택합니다. 그런 다음 기존 서브넷 그룹을 선택하거나 새 서브넷 그룹을 생성합니다.

네트워크 유형

IPv4, 듀얼 스택 및 IPv6 중에서 선택

IPv4



ElastiCache

이 클러스터의 샷드 수 입력(1~500개)

3

샤드당 복제본 수

각 샷드에 대한 복제본 수(0~5개)를 입력합니다.

1

연결

이 클러스터가 지원할 IP 버전을 선택합니다. 그런 다음 기존 서브넷 그룹을 선택하거나 새 서브넷 그룹을 생성합니다.

네트워크 유형

IPv4, 둑일 스택 및 IPv6 중에서 선택

IPv4

리소스는 IPv4 프로토콜을 통해서만 통신합니다.

서브넷 그룹

기존 서브넷 그룹 선택

새 서브넷 그룹 생성

서브넷 그룹

Amazon VPC에서 실행 중인 클러스터에 대해 지정할 수 있는 서브넷 모음입니다.

qwer (vpc-05ec4188994eb7d90)



연결된 서브넷 (4)



가용 영역	▲	서브넷 ID	▼	CIDR 블록(IPv4)	▼
ap-northeast-2a		subnet-015e294a8319211cb		172.31.0.0/20	
ap-northeast-2b		subnet-006d5510bad85195f		172.31.16.0/20	
ap-northeast-2c		subnet-0c9dc86804732fc1b		172.31.32.0/20	
ap-northeast-2d		subnet-02cf16a38e8c199a1		172.31.48.0/20	

가용 영역 배치

다음 필드를 사용하여 가용 영역에 대한 배치를 구성합니다.

슬롯 및 키스페이스



ElastiCache

연결된 서브넷 (4)

가용 영역	서브넷 ID	CIDR 블록(IPv4)
ap-northeast-2a	subnet-015e294a8319211cb	172.31.0.0/20
ap-northeast-2b	subnet-006d5510bad85195f	172.31.16.0/20
ap-northeast-2c	subnet-0c9dc8680473fc1b	172.31.32.0/20
ap-northeast-2d	subnet-02cf16a38e8c199a1	172.31.48.0/20

가용 영역 배치

다음 필드를 사용하여 가용 영역에 대한 배치를 구성합니다.

슬롯 및 키스페이스

샤드에 걸친 16,384개의 Redis 클러스터 키스페이스 슬롯의 배포입니다.

사용자 지정 배포

가용 영역 배치

다른 가용 영역에 노드를 배치하여 한 가용 영역에 정전 같은 장애가 발생할 경우 전체 시스템에 장애가 발생할 가능성을 줄일 수 있습니다. 클러스터 노드에 가용 영역을 지정하려면 **Specify Availability Zones(가용 영역 지정)**을 선택합니다.

가용 영역 지정

샤드			슬롯/키스페이스	프라이머리	복제본 1
샤드 1	0-5000		ap-northeast-2a ▾	ap-northeast-2b ▾	
샤드 2	5001-10000		ap-northeast-2a ▾	ap-northeast-2b ▾	
샤드 3	10001-16383		ap-northeast-2a ▾	ap-northeast-2b ▾	

취소

다음



의견

개인 정보 보호 약관 쿠키 기본 설정

© 2024, Amazon Web Services, Inc. 또는 계열사.



ElastiCache

클러스터 모드 비활성화

The screenshot shows the AWS ElastiCache console for a cluster named 'qwer'. It displays various configuration settings:

- 사드: 1
- 노드 수: 3
- 다중 AZ 활성화됨
- 자동 장애 조치 활성화됨
- 전송 중 암호화 비활성화됨
- 파라미터 그룹: default.redis7
- Outpost ARN: -
- 기본 엔드포인트: qwer.sph4da.ng.0001.apn2.cache.amazonaws.com:6379
- 리더 엔드포인트: qwer-ro.sph4da.ng.0001.apn2.cache.amazonaws.com:6379
- ARN: arn:aws:elasticache:ap-northeast-2:404391069370:replicationgroup:qwer
- 데이터 마이그레이션: 활성 마이그레이션 없음

At the bottom, there are CloudShell and 의견 (Feedback) buttons.

The screenshot shows the '노드' (Nodes) page for the 'qwer' cluster. It lists three nodes:

노드 이름	상태	현재 역할	엔드포인트
qwer-001	Available	primary	qwer-001
qwer-002	Available	replica	qwer-002
qwer-003	Available	replica	qwer-003

At the bottom, there are CloudShell and 의견 (Feedback) buttons.

기본 엔드포인트

리더 엔드포인트

노드 엔드포인트



ElastiCache

클러스터 모드 활성화

The screenshot shows the AWS ElastiCache console for a cluster named 'qwer'. It displays various configuration parameters:

설정	값
샤드	2
자동 장애 조치 활성화됨	활성화됨
전송 중 암호화 비활성화됨	저장 중 암호화 비활성화됨
파라미터 그룹 <code>default.redis7.cluster.on</code>	Outpost ARN -
구성 엔드포인트 qwer.sph4da.clustercfg.apn2.cache.amazonaws.com:6379	기본 엔드포인트 -
리더 엔드포인트	ARN arn:aws:elasticache:ap-northeast-2:404391069370:replicationgroup:qwer
데이터 마이그레이션	활성 마이그레이션 없음

At the bottom, there are links for CloudShell and的意见 (Feedback), and standard footer links for 개인정보 보호, 약관, and 쿠키 기본 설정.

The screenshot shows the AWS ElastiCache console displaying node endpoints for the 'qwer' cluster. The table lists nodes grouped by shard ID (0-8000 and 8001-16383) and their specific ARNs.

이름	유형	샤드당 노드 수	슬롯/키스페이스	영역
qwer-0001	샤드	3	0-8000	-
qwer-0001-001	노드	-	-	ap-northeast-
qwer-0001-002	노드	-	-	ap-northeast-
qwer-0001-003	노드	-	-	ap-northeast-
qwer-0002	샤드	3	8001-16383	-
qwer-0002-001	노드	-	-	ap-northeast-
qwer-0002-002	노드	-	-	ap-northeast-
qwer-0002-003	노드	-	-	ap-northeast-

At the bottom, there are links for CloudShell and 의견 (Feedback), and standard footer links for 개인정보 보호, 약관, and 쿠키 기본 설정.

구성 엔드포인트

노드 엔드포인트

Content

**Cache
Redis
ElastiCache**



Thank You

이상 인하대학교 ACC 가동식이었습니다.

Info. 기동식

EMAIL: ret0422@gmail.com

Github: <https://github.com/Nesquitto>

LinkedIn: www.linkedin.com/in/Nesquitto