분산 환경에서 스트림과 배치 처리 통합 모델

DEVIEW
2015

정재부 이도경
네이버 검색정제개발랩

NAVER

contents

1.
배치 처리 vs. 스트림 처리

배치 처리 vs. 스트림 처리

배치 처리(Batch Processing)

배치 처리 vs. 스트림 처리

배치 처리(Batch Processing)

Hadoop Mapreduce, Spark…

배치 처리 vs. 스트림 처리

스트림 처리(Stream Processing)

배치 처리 vs. 스트림 처리

스트림 처리(Stream Processing)

Storm, Flink, Samza…

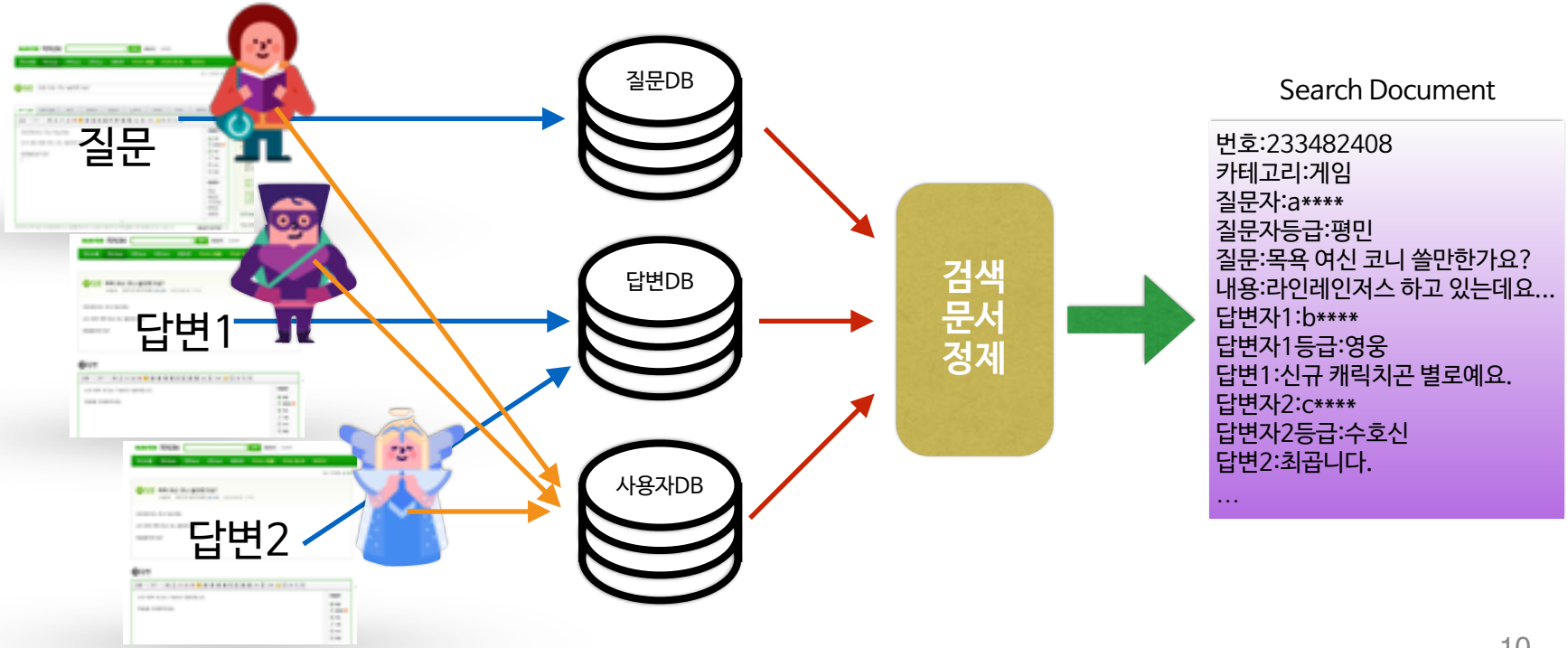
배치 처리 vs. 스트림 처리

High Throughput vs. Low Latency

배치 처리 vs. 스트림 처리

검색 문서 정제란?

정규화된 데이터를 검색에 적합한 형태로 가공하는 작업



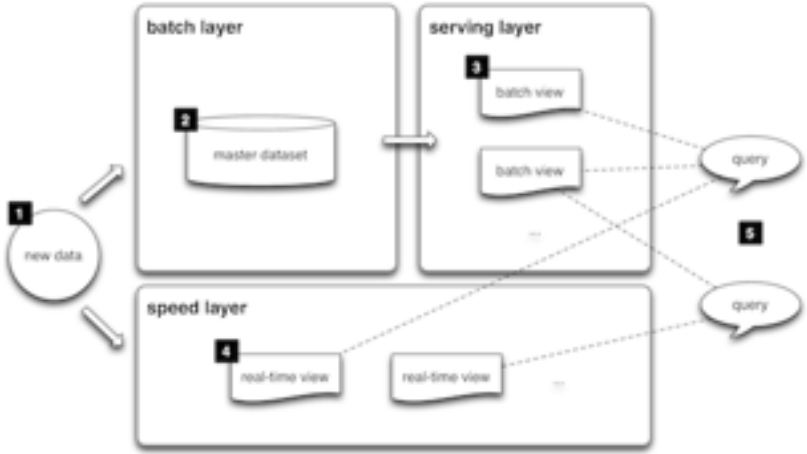
검색 문서 정제란?

요구 사항

1. 많은 검색 대상 문서 처리가 가능해야함(High Throughput)
2. 최신 문서를 빠르게 검색에 노출(Low Latency)
3. 서비스별 비지니스 로직 적용

스트림과 배치 처리 통합 모델 사례

Lambda Architecture

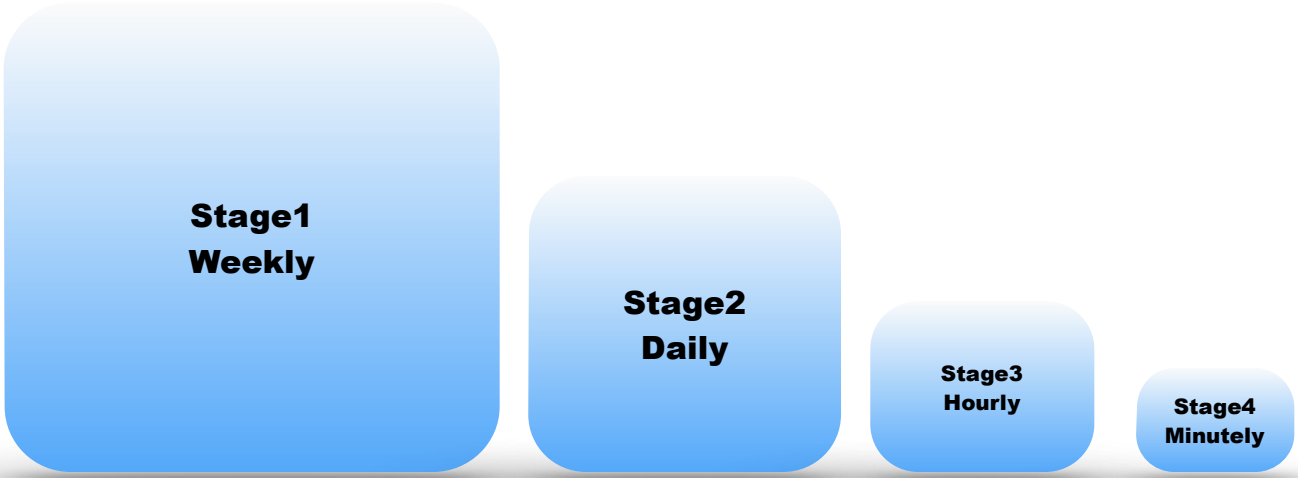


http://lambda-architecture.net/

기존 검색 문서 정제 방식

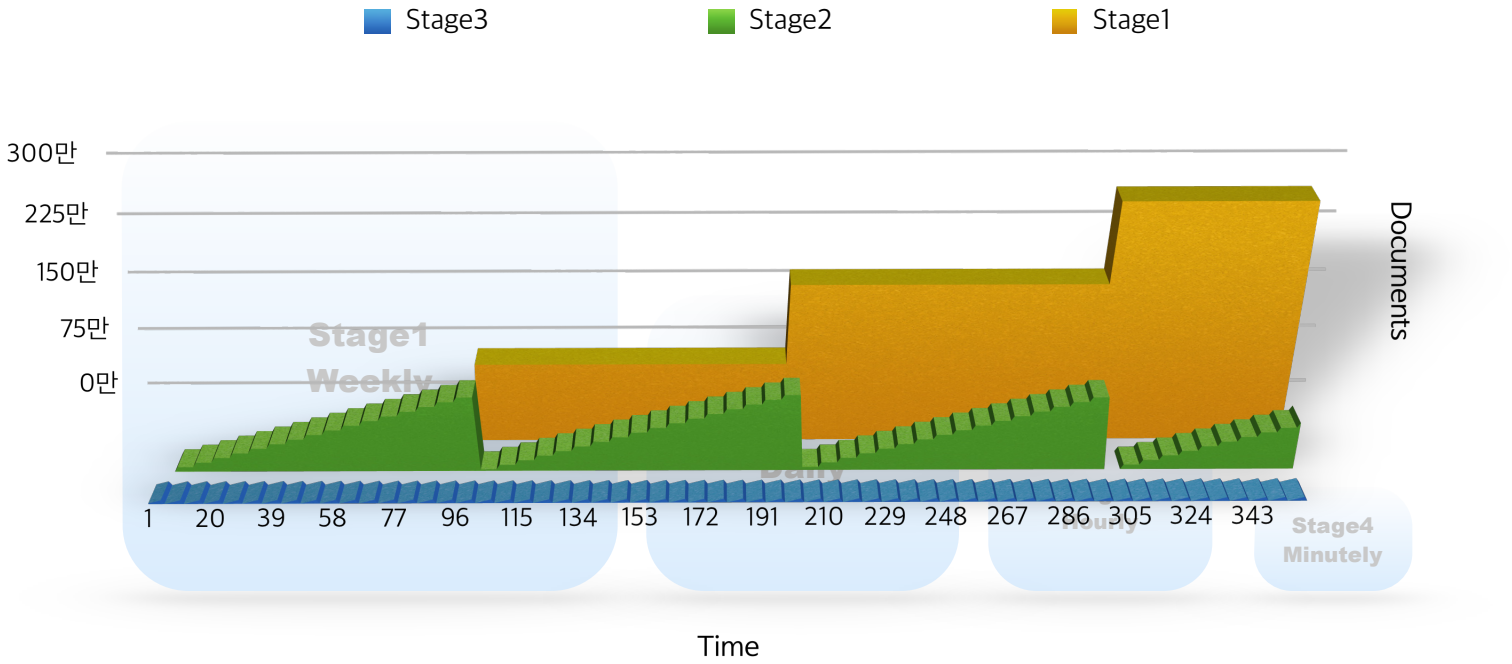
Timewise Staged 배치 시스템

예) Stage1은 1주일마다, Stage2는 1일마다, Stage3는 1시간마다, Stage4는 1분마다



기존 검색 문서 정제 방식

Timewise Staged 배치 시스템



Timewise Staged 배치 시스템의 장점

1. 배치 처리만으로 Low Latency 만족 가능
   - ➡Latency 요구 사항을 만족하지못하면 Stage를 추가하면 된다.

2. 정제 로직 변경이 단순
   - ➡모든 데이터는 결국 다시 처리

Timewise Staged 배치 시스템의 단점

DEVIEW 2015

1. Stage간에 부정확성이 발생

   ➡상위 Stage가 재수행되면 해결

2. 주기적인 전체 데이터 재처리 수행

어떻게하면 전체 재처리를 피할까?

증분 시스템(Incremental System)을 만들자.

$$A_{n+1} = A_n + n$$

네이버의 검색 증분화 프로젝트(**BREW Project**)

➡2015년 상반기 첫 검색 서비스 적용

증분 시스템을 만들려면?

1. Random Read/Write Storage
2. Transaction
3. Stream(Event-driven) Processing

증분 시스템의 특징

장점

- ➡주기적인 전체 재처리 X
- ➡Low Latency

증분 시스템의 특징

단점

- ➡ Persistent Storage Maintenance Hell
- ➡ 배치 처리 로직과 스트림 처리 로직의 통합은 아직...

증분 시스템은 어떻게 만들지?

정제 로직에만 집중할 수 없을까?

1. 너무 많은 솔루션(필요한데...)
2. 너무 많은 서비스(적용하기엔...)

결국 Abstraction이 답!

➡ Programming Model Abstraction

2.
Domaine Architecture

Domaine Architecture

주요 특징

- Transaction, Data Load/Sink Code 숨기기
- Event: No Data, No action
- Diff: previous/current value 제공

Domaine Architecture

공간 제약이 없는

메모리가 있고

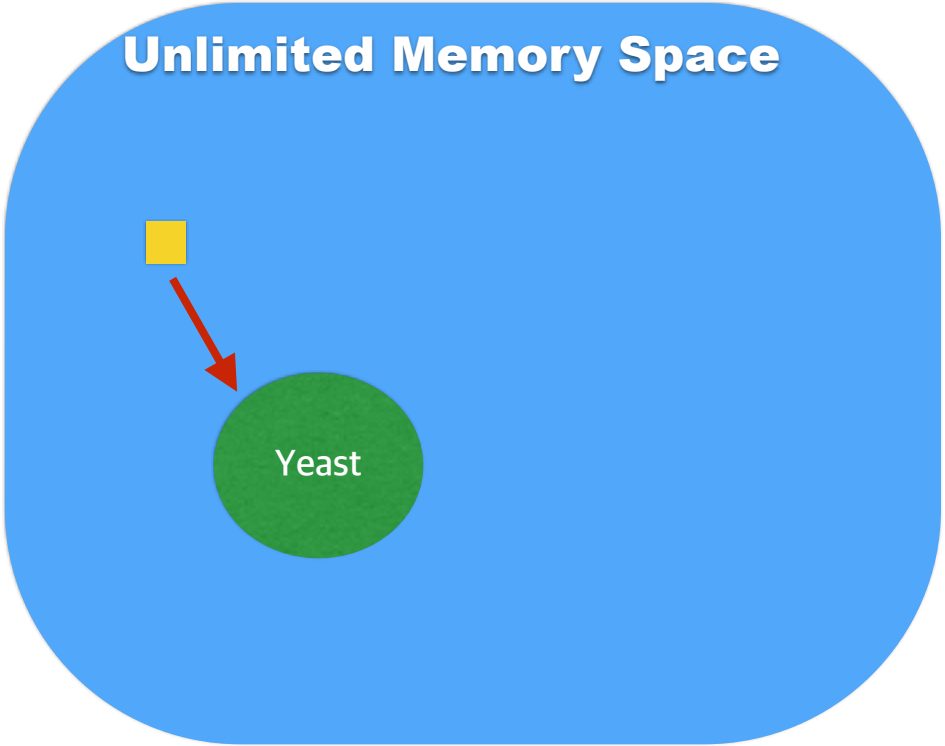
Domaine Architecture

메모리상의

어떤 Cell의 값이 변하면

Domaine Architecture

그 Cell에 관심이 있던
Yeast가 실행된다.

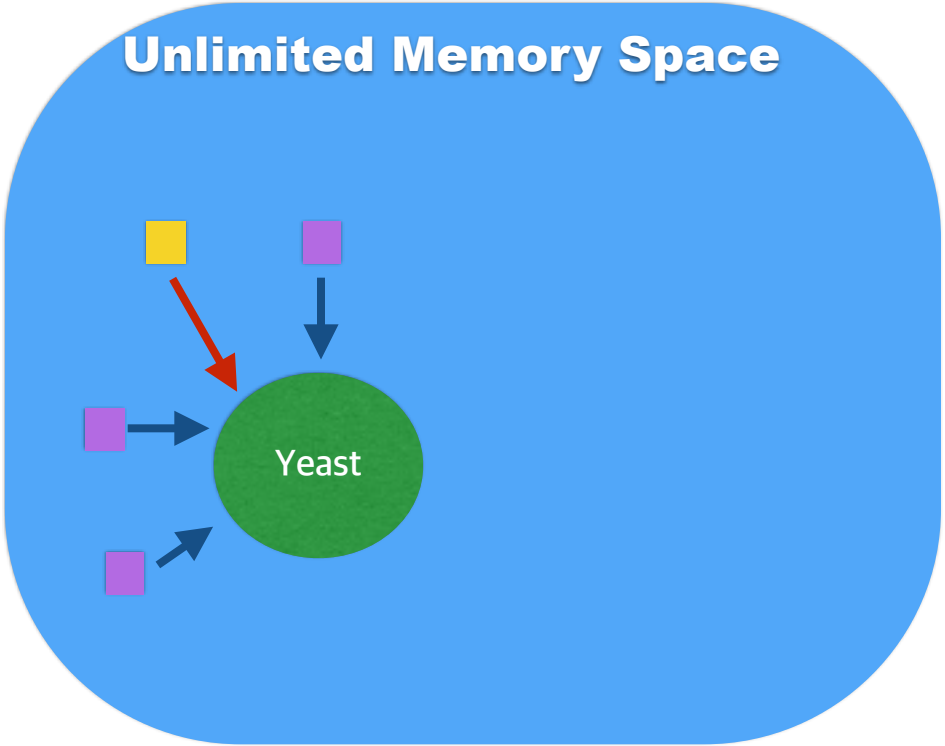


Domaine Architecture

Yeast는 Process Unit이고

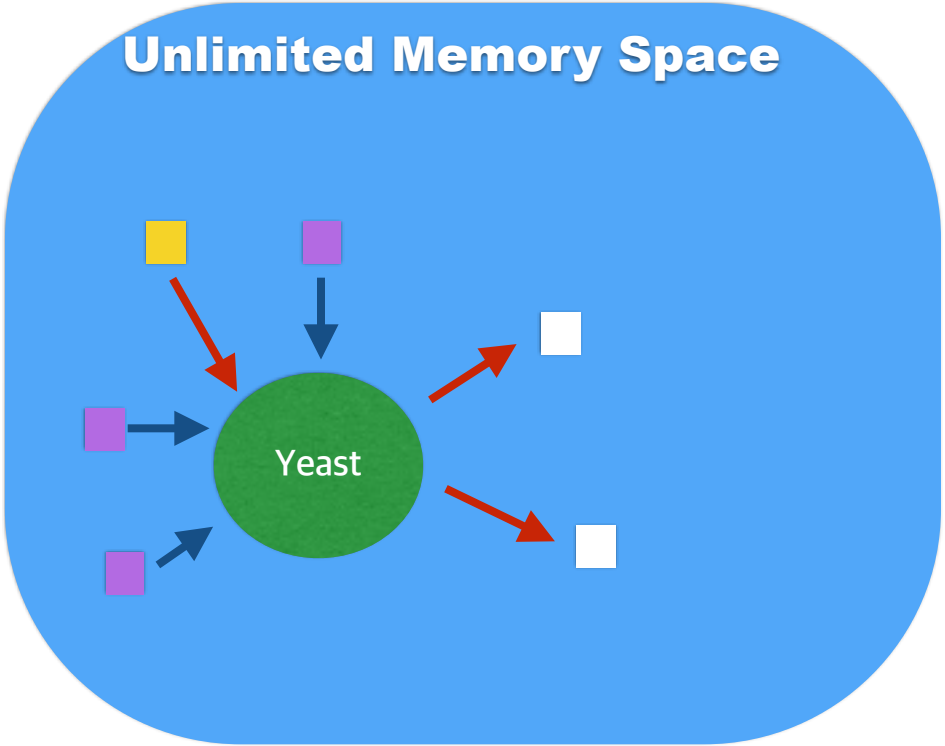
실행되면 로직에 따라

필요한 값을 읽어서



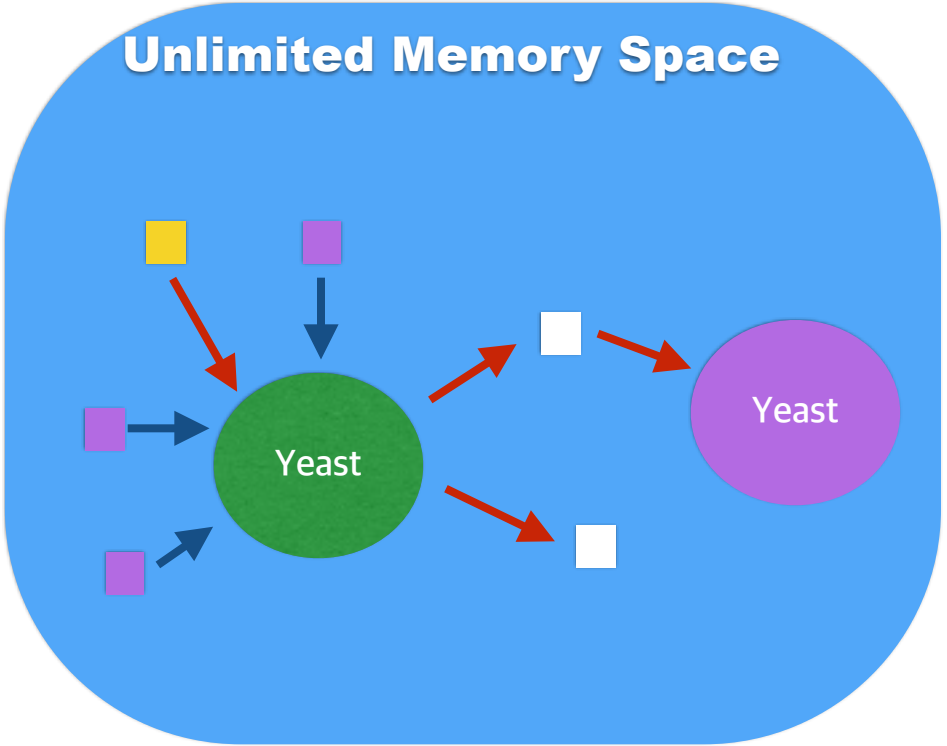
Domaine Architecture

결과를 메모리에 반영한다.



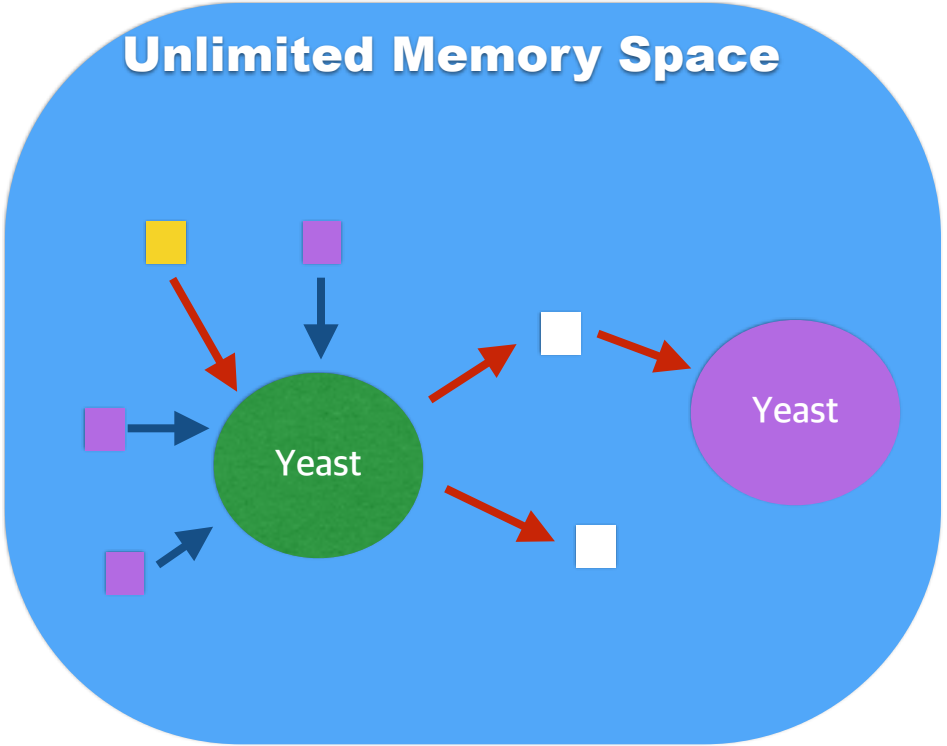
Domaine Architecture

새로 반영된 값에
관심이 있는
다른 Yeast가 있으면
해당 Yeast가 실행된다.



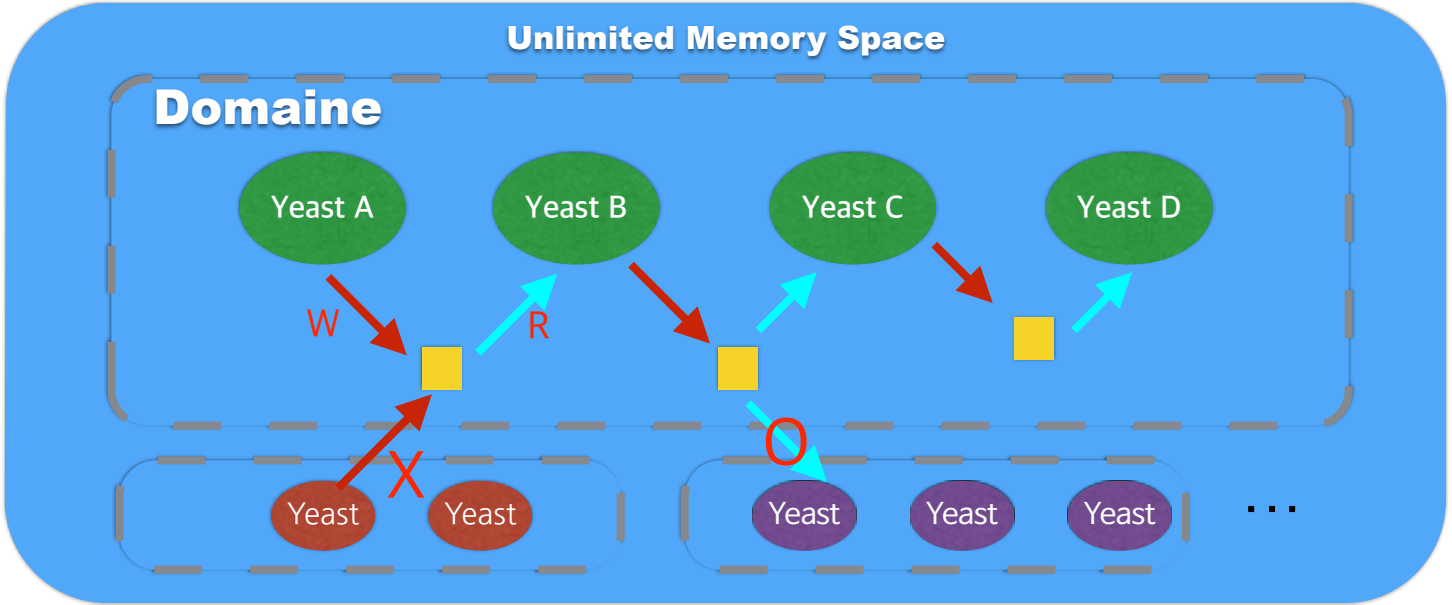
Domaine Architecture

Yeast 하나가
실행되는 동안에는
논리적으로 전체 메모리를
혼자서만 바꿀수 있다.



Domaine Architecture

Yeast와 메모리는 Domaine이라는 영역(Scope)으로 한정
같은 Domaine내 메모리는 R/W가 가능, 다른 Domaine은 R만 가능



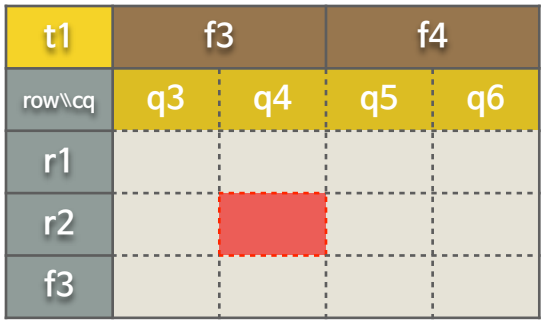
Domaine Architecture

오직 메모리에서 읽어서 처리후 다시 메모리에 쓴다.

Software Transactional Memory와 유사

```
Cell c1 = domaine.table("t1").row("r2").family("f3").qualifier("q4");
```

```
int a = c1.getInt(0);
c1.set(a + 1);
```

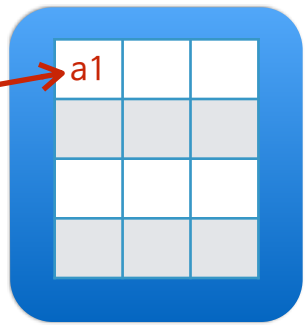


Domaine Development Kit

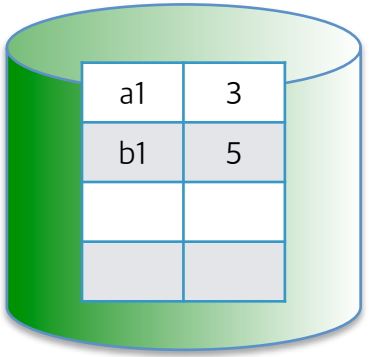
DEVIEW 2015

```
int a = a1.getInt(0);
```

miss!



StorageSandbox



Physical Storage

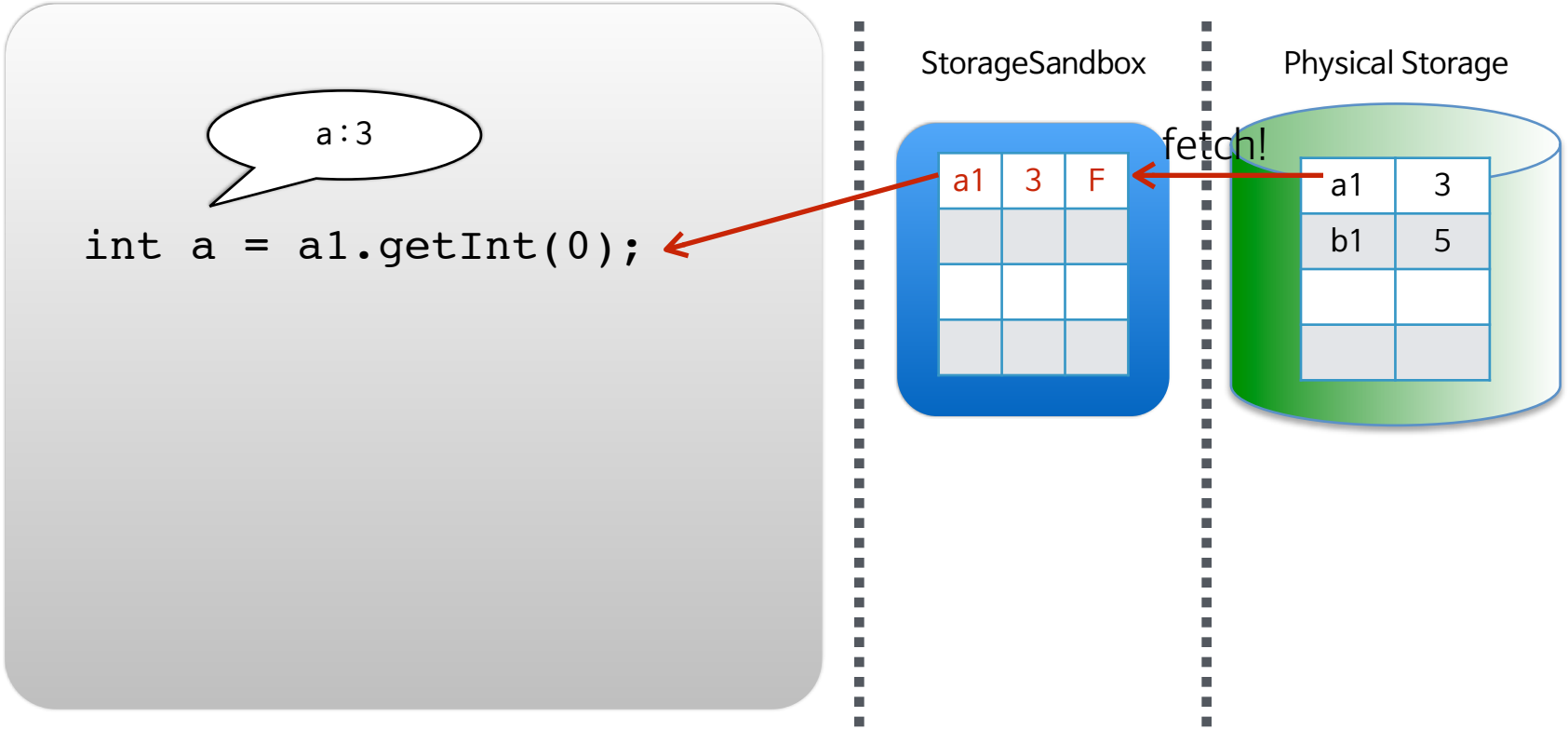
Yeast Layer

DDK Layer

Storage Layer

Domaine Development Kit

DEVIEW 2015



Domaine Development Kit

DEVIEW 2015

```
int a                nt(0);

int b = a1.getInt(1);
```

b:3

StorageSandbox

a1	3	F

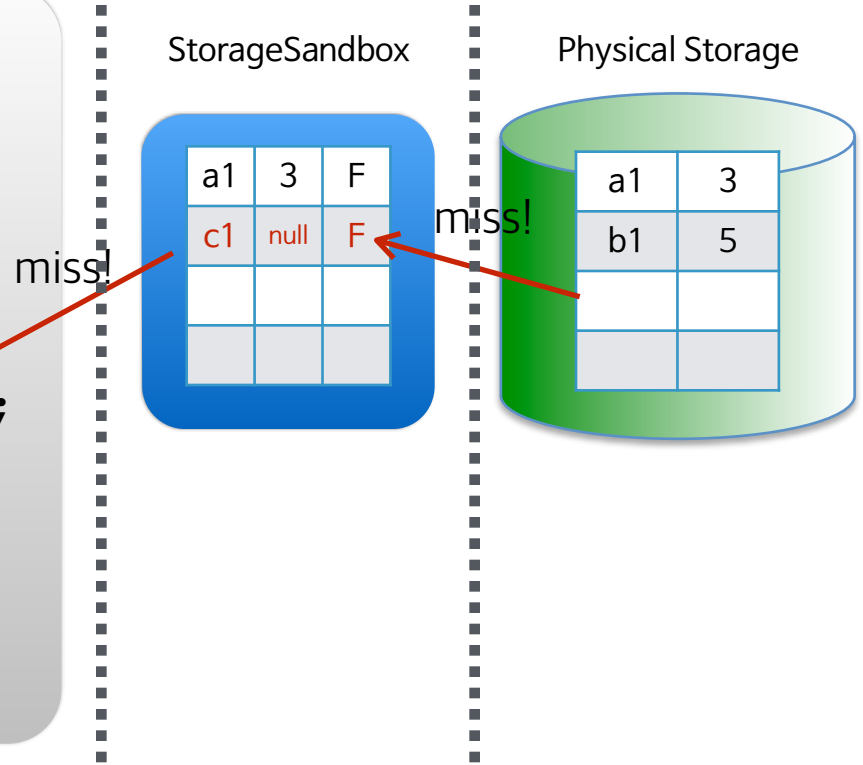
Physical Storage

a1	3
b1	5

Domaine Development Kit

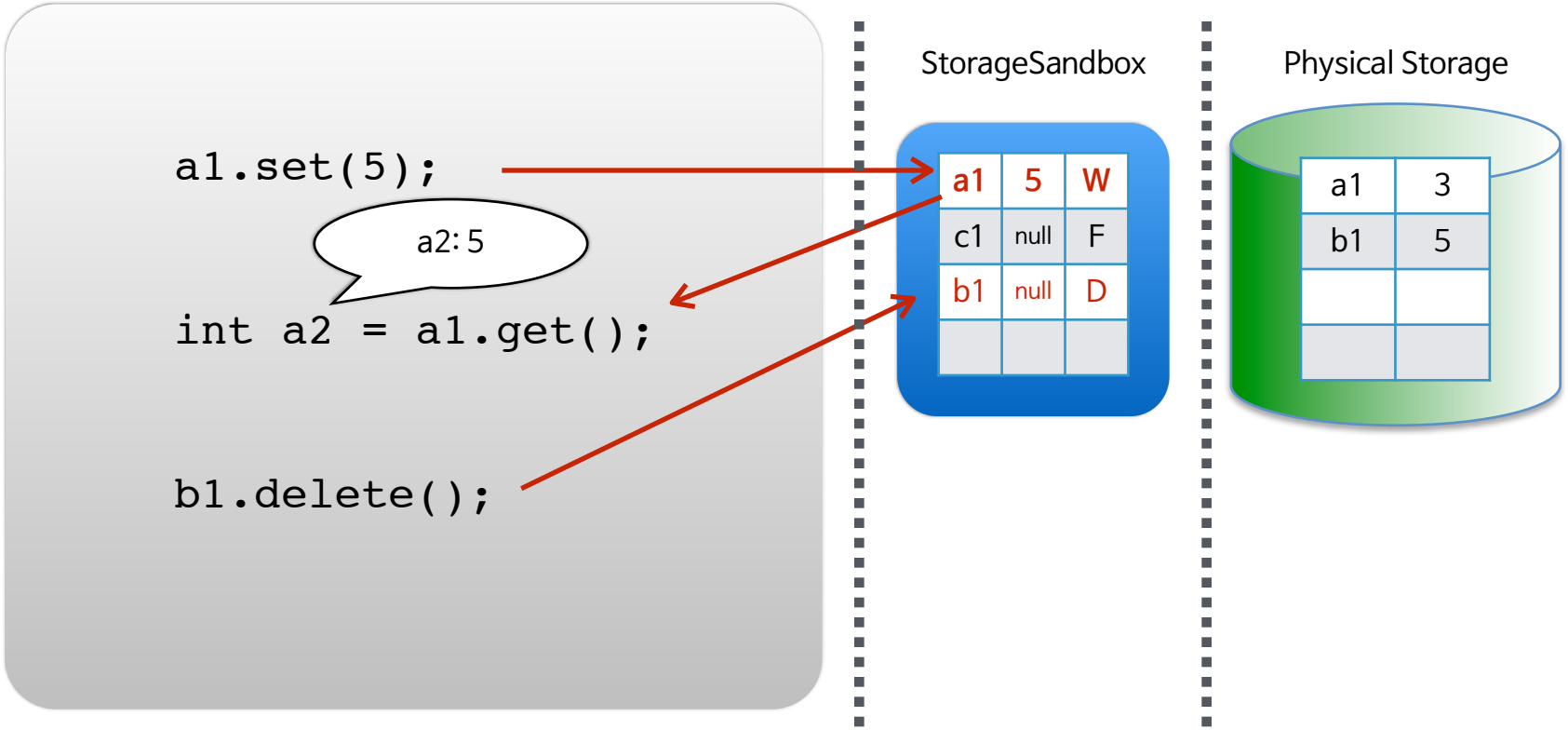

```
int a = a1.getInt(0);

int b = b1.getInt(1);

int c = c1.getString("c");
```

c: "c"



Domaine Development Kit

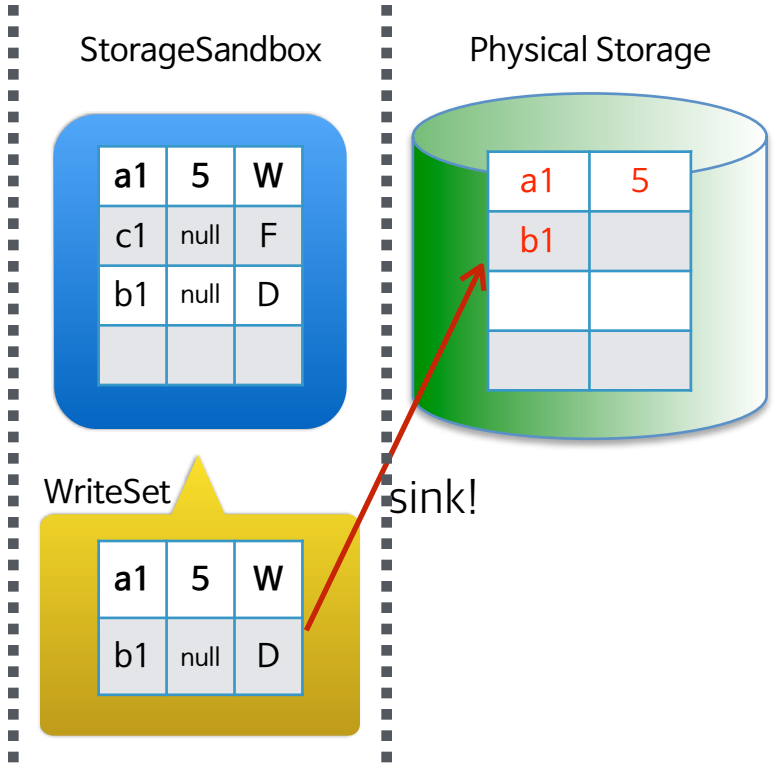
DEVIEW 2015



Domaine Development Kit

DEVIEW 2015

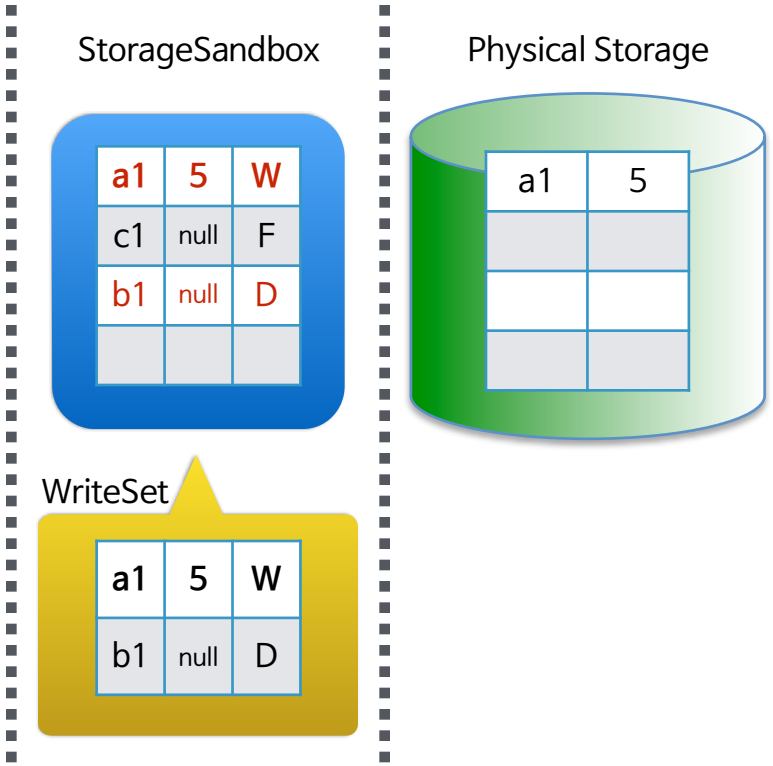
```
a1.set(5);

int a2 = a1.get();

b1.delete();
```



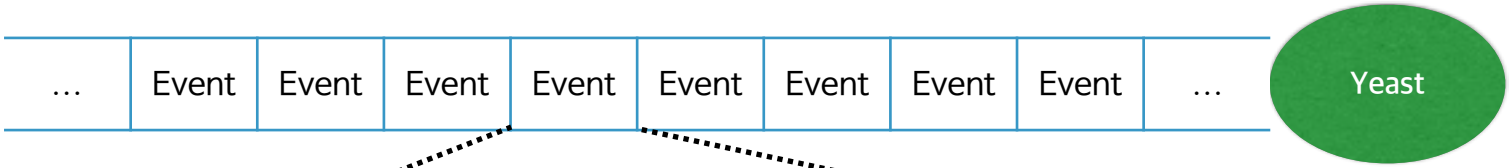
Domaine Development Kit

DEVIEW 2015

```
a1.set(5);

int a2 = a1.get();

b1.delete();
```

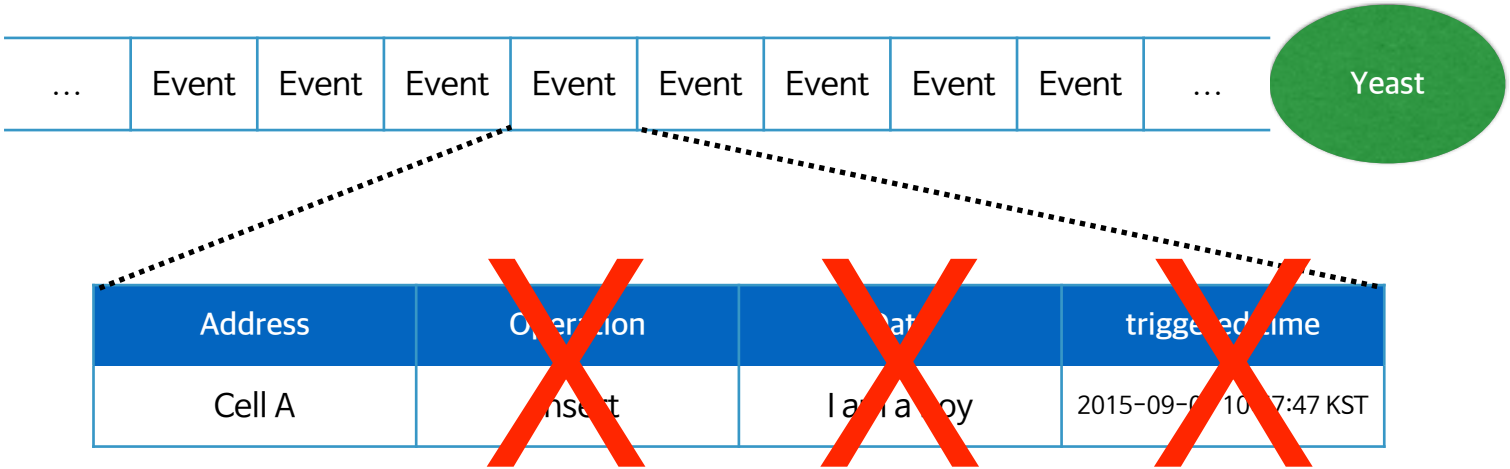


일반적인 Event Message



Address	Operator	Data	triggered time
Cell A	Insert	I am a boy	2015-09-07 10:57:47 KST

Yeast Event



Event에는 변한 메모리 주소만 있을 뿐 Operator와 Data가 없다.

➡ 데이터 순서 역전 방지 및 최신성 문제를 해결

➡ Operator는 Yeast가 직접 결정

Yeast Event

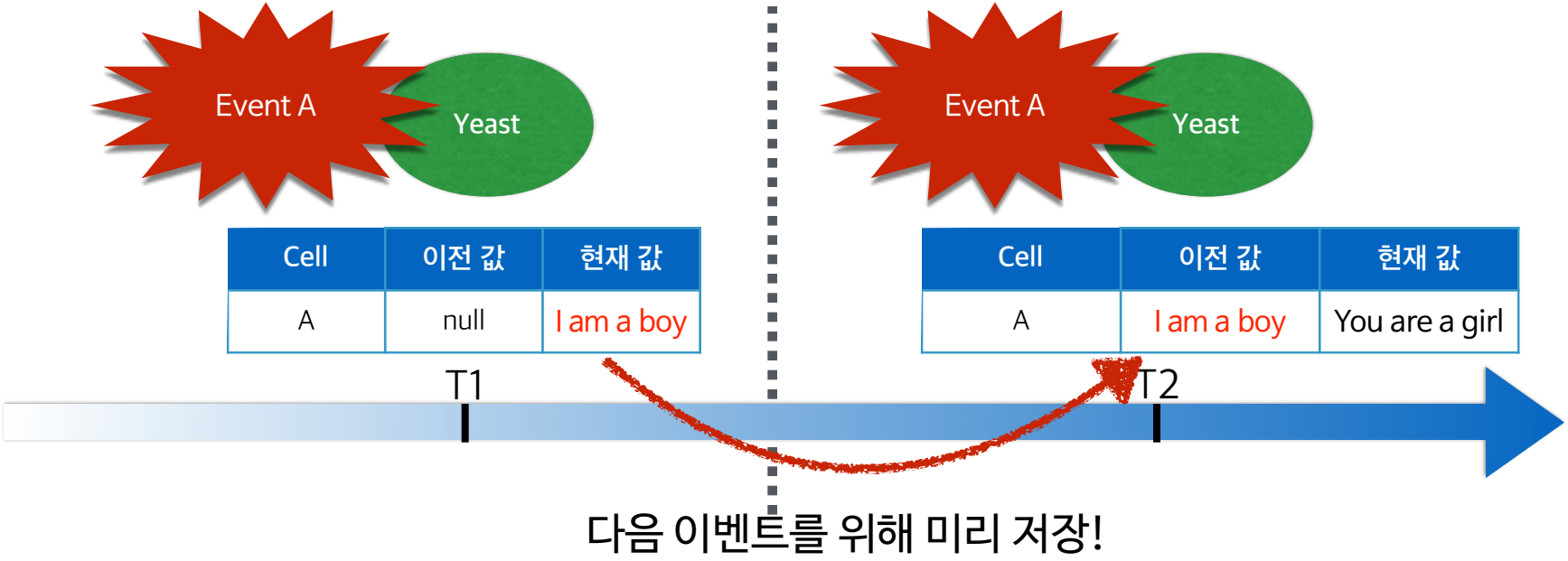
Yeast에서 관심있던 Cell에 이전 값과 현재 값을 알 수 있다.

➡ Operator로 사용

이전 값	현재 값	Operator
null	A	Insert
A	B	Update
B	null	Delete
C	C	NoAction

Diff

Event Cell 이전값/현재값 제공 방법



3.
Domaine Development Kit Example

Example Code


```
package com.naver.kessel.ddk;

import ...

public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT, table = TB_SOURCE, family = CF_SOURCE, qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT).family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord : target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word : target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Example Code

관심있는 메모리 영역을 표시

```
package com.naver.kessel.ddk;

import ...

public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT, table = TB_SOURCE, family = CF_SOURCE, qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT).family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord : target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word : target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Example Code


```
package com.naver.kessel.ddk;

import ...

public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT, table = TB_SOURCE, family = CF_SOURCE, qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT).family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord : target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word : target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Diff를 이용한 Extactly-Once 처리

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Example Code

DEVIEW 2015

```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am a boy

Result	
Row	Count
I	1
am	1
a	1
boy	1

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am a boy
document2	You are a girl

Result	
Row	Count
I	1
am	1
a	2
boy	1
you	1
are	1
girl	1

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am not a girl
document2	You are a girl

Result	
Row	Count
I	1
am	1
a	2
boy	1
you	1
are	1
girl	1

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am not a girl
document2	You are a girl

Result	
Row	Count
I	1 - 1
am	1 - 1
a	2 - 1
boy	1 - 1
you	1
are	1
girl	1

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am not a girl
document2	You are a girl

Result	
Row	Count
I	0 + 1
am	0 + 1
a	1 + 1
boy	0
you	1
are	1
girl	1 + 1
not	1

Example Code


```
public class WordCountYeast implements Yeast {

    @Interest(domaine = DM_WORDCOUNT,
            table = TB_SOURCE,
            family = CF_SOURCE,
            qualify = CQ_SENTENCE)
    public int wordcount(Domaine domaine, InterestingCell target) {

        if (!target.isChanged()) {
            return SKIP;
        }

        CellWithoutRow cwr = domaine.table(TB_RESULT)
                .family(CF_RESULT).qualify(CQ_COUNT);

        for (String prevWord :
                target.getPreviousString("").split(" ")) {
            cwr.row(prevWord).increase(-1);
        }

        for (String word :
                target.getString("").split(" ")) {
            cwr.row(word).increase(1);
        }

        return OK;
    }
}
```

Source	
Row	Sentence
document1	I am not a girl
document2	You are a girl

Result	
Row	Count
I	1
am	1
a	2
boy	0
you	1
are	1
girl	2
not	1

4.
초기 구축 배치 처리 방법

기존 데이터는 어떻게 처리하지?

현재 들어오는 입력은 스트림으로 처리
그러면 이전 데이터도 스트림으로?

➡ 스트림으로 처리하기에는 양이 너무 많다.

처리량을 높이기 위해서는 배치 처리가 필요

➡ 스트림 로직과 별개로 배치 로직을 작성해줘야 하나?

기존 데이터는 어떻게 처리하지?

Summingbird

배치 모드와 실시간 모드에서 프로그램 수행 가능
➡ 로직을 두번 작성할 필요 없음

같은 키를 갖는 값들에 대한 monoid 연산 제공
➡검색 로직 작성이 어려운 경우가 있음

https://github.com/twitter/summingbird

Yeast를 배치로 처리하려면?

그냥 Yeast 를 배치로 처리해보면 어떨까?

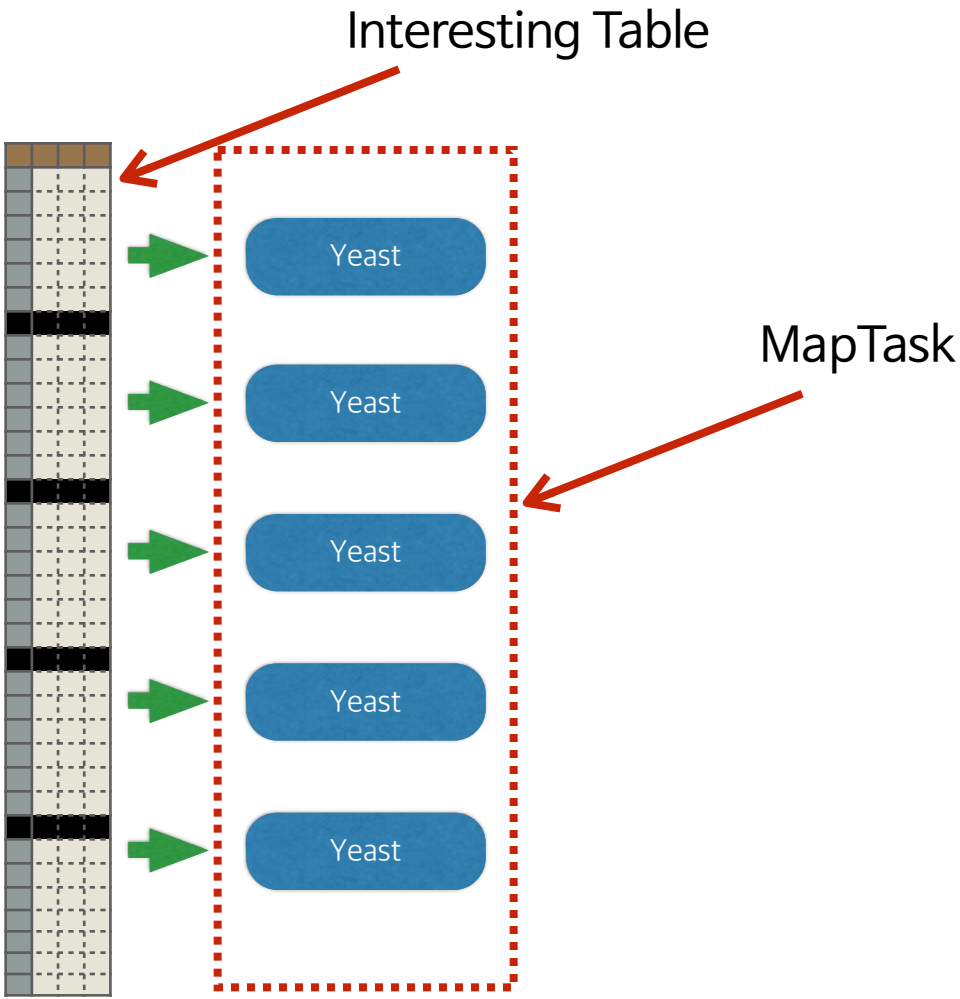
- ➡ Yeast는 트랜잭션으로 묶여 있으니 무수히 많은 conflict가 발생할텐데?
- ➡ Yeast 내에서 Storage에 대한 Random Read가 수행되는데, 처리량이 만족스러울까?

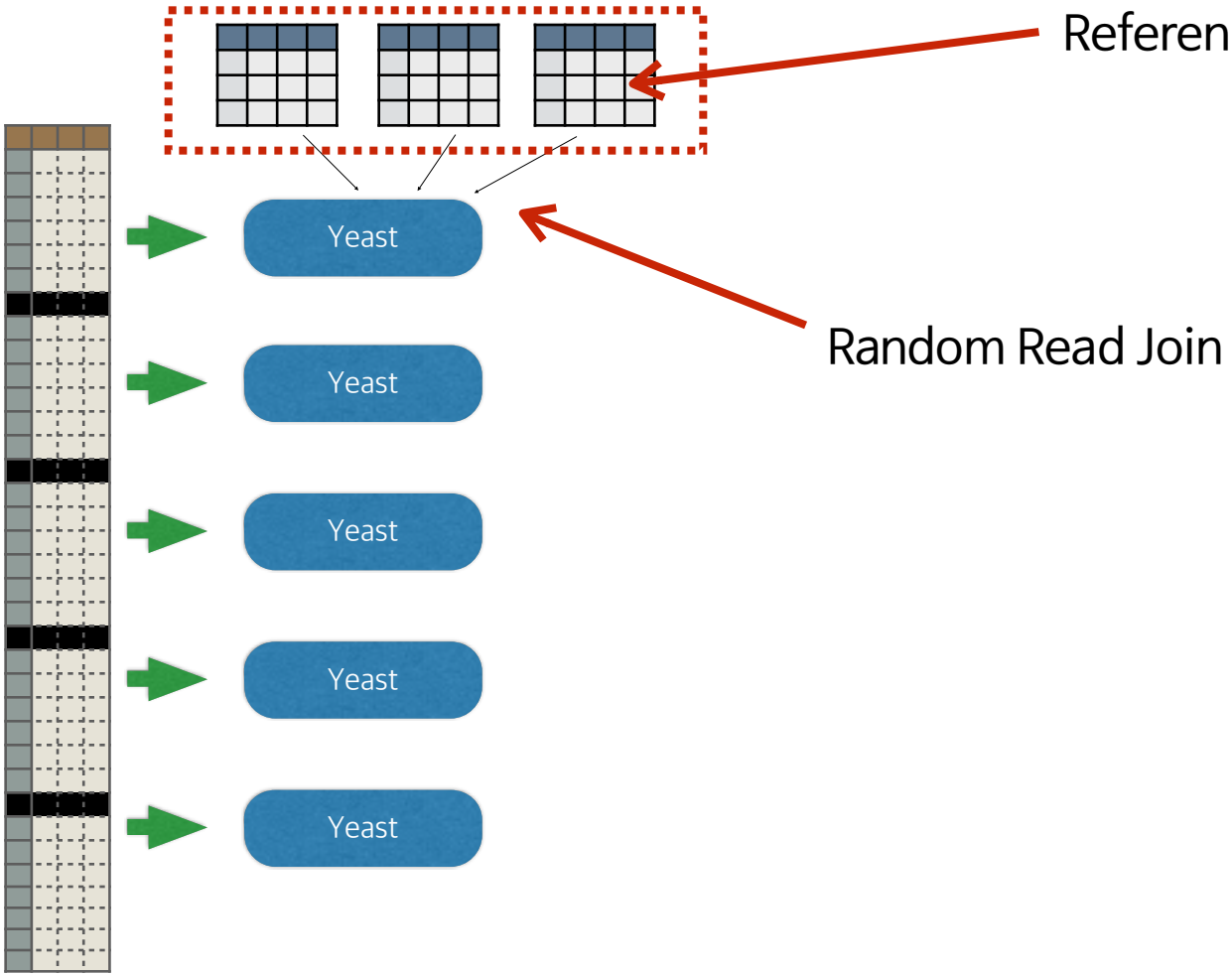
Yeast 배치 처리 중 Conflict 체크

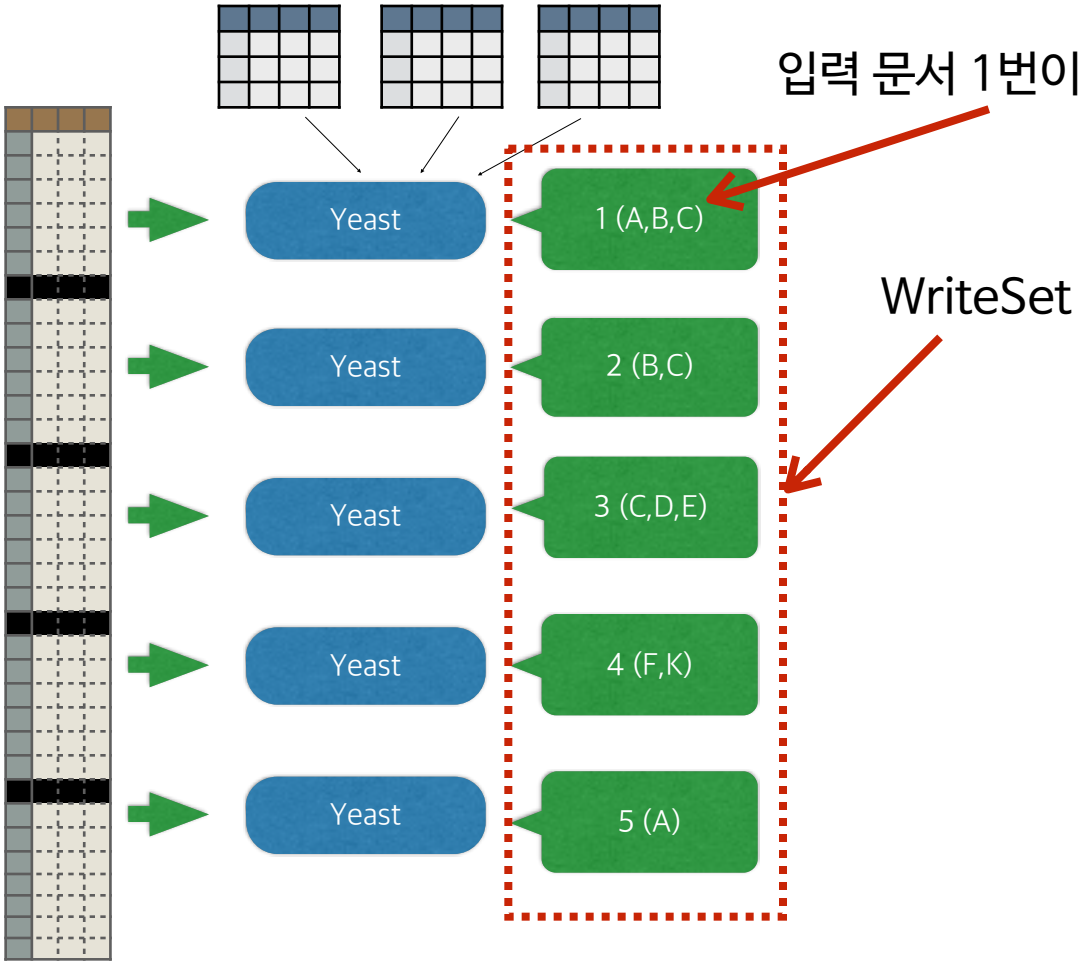
배치에서는 write-write conflict 만 체크하면 된다.

- ➡ 사용하는 데이터는 특정 시점의 storage snapshot 내용이기 때문에 write한 데이터를 입력으로 사용하지 않음
- ➡ Yeast 결과인 write set이 같은 경우가 있는지 체크

 Optimistic Concurrency Control

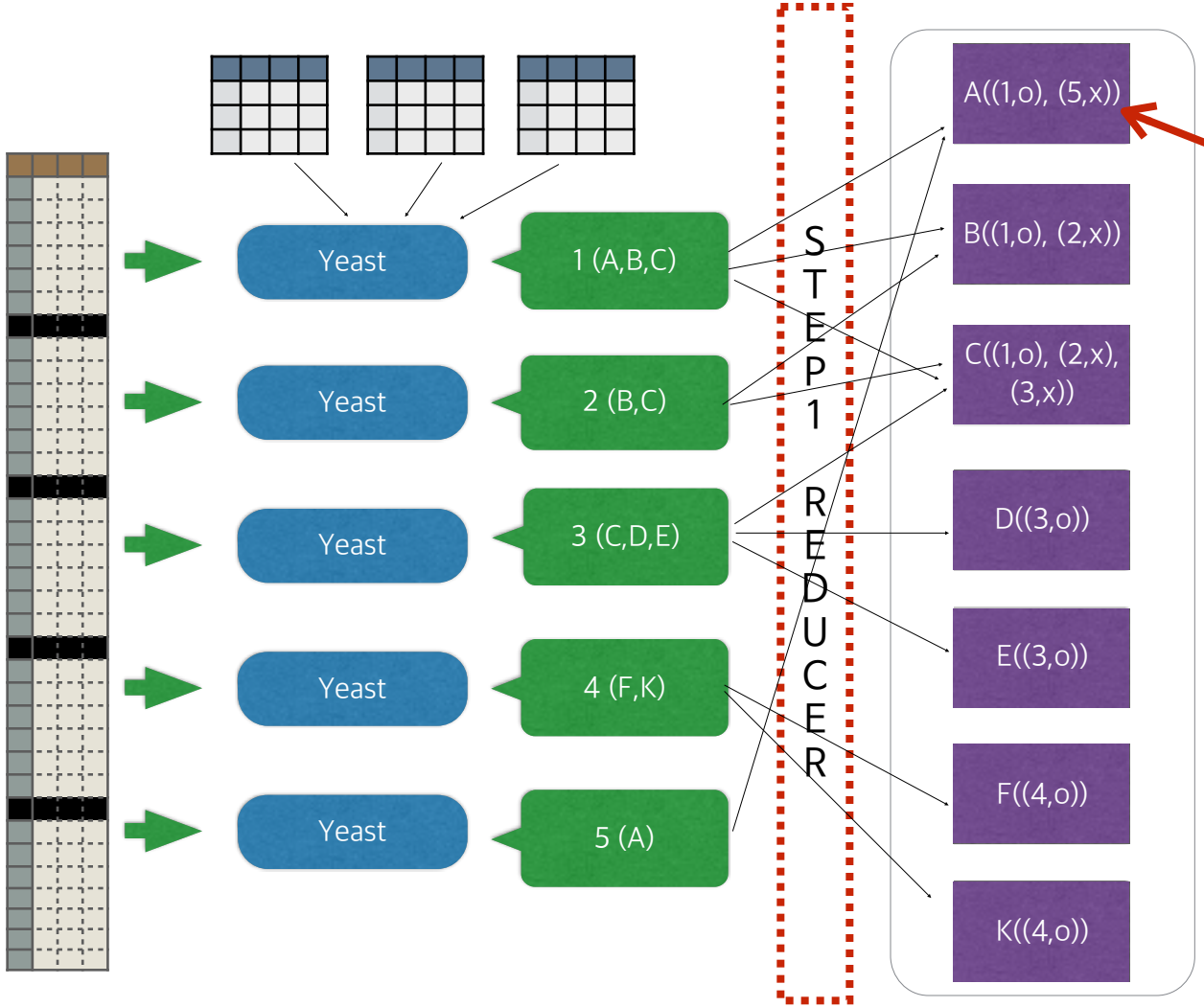

Interesting Table
Yeast
Yeast
Yeast
Yeast
Yeast
MapTask


Reference Table
Yeast
Yeast
Yeast
Yeast
Yeast
Random Read Join

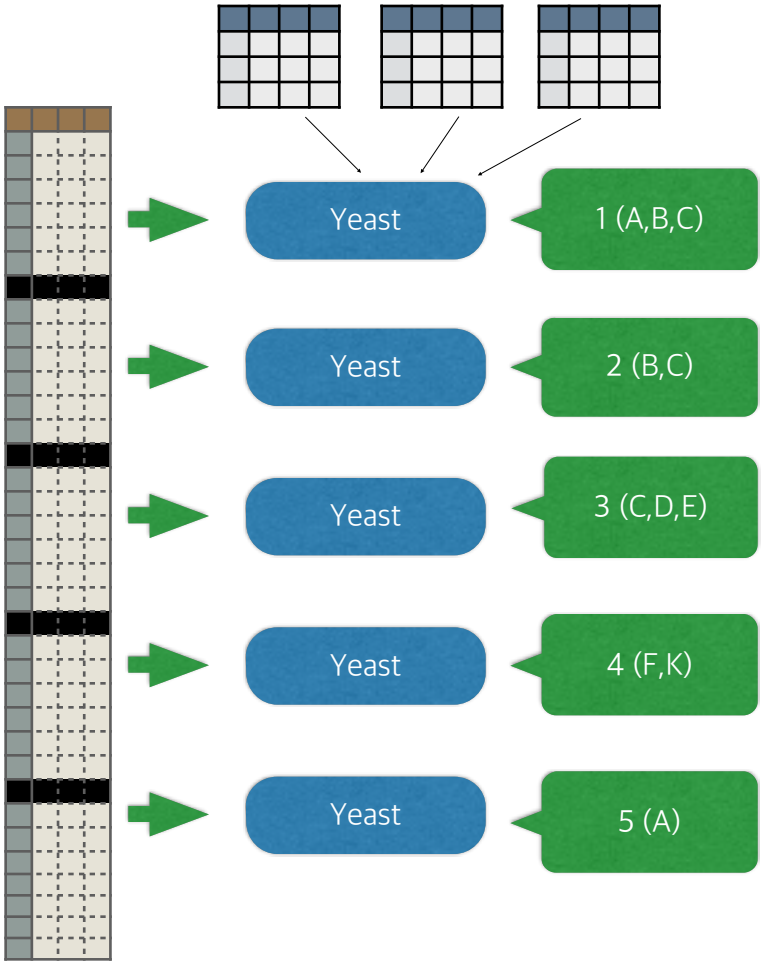


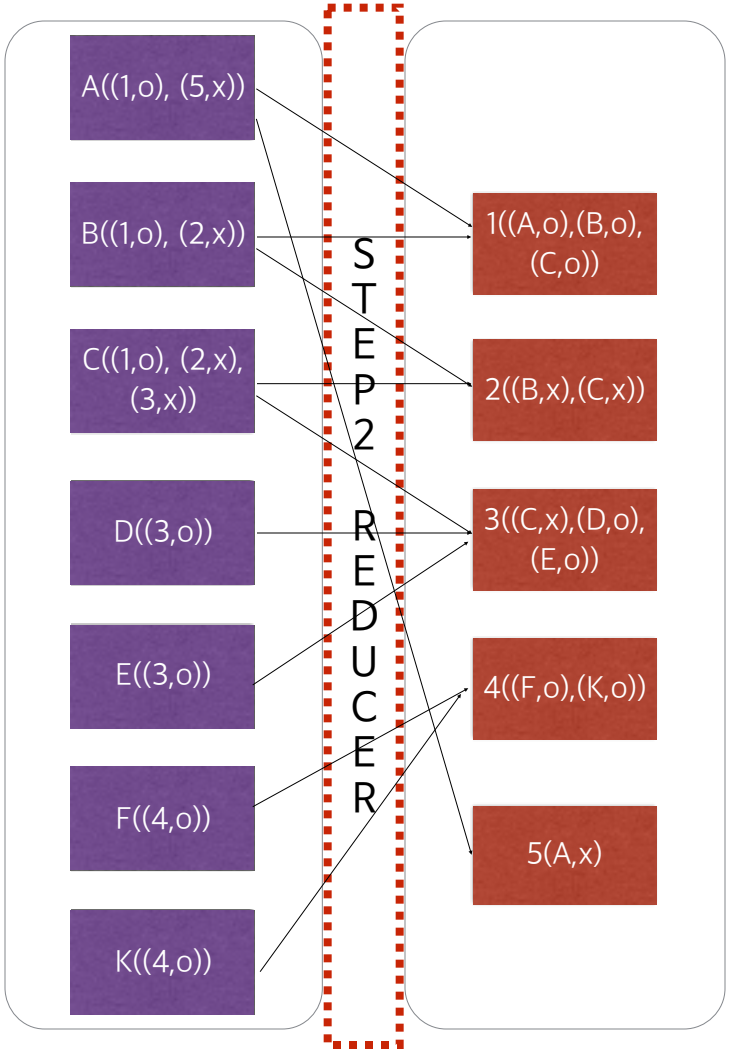
입력 문서 1번이 cell A,B,C 에 결과 값을 썼다.

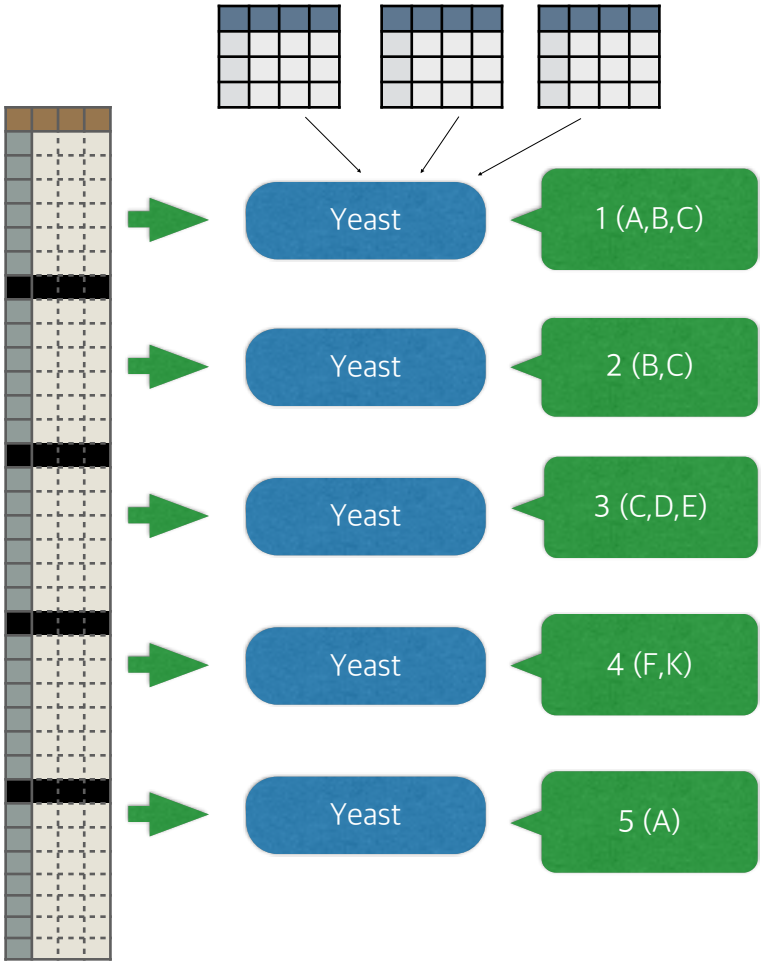
WriteSet

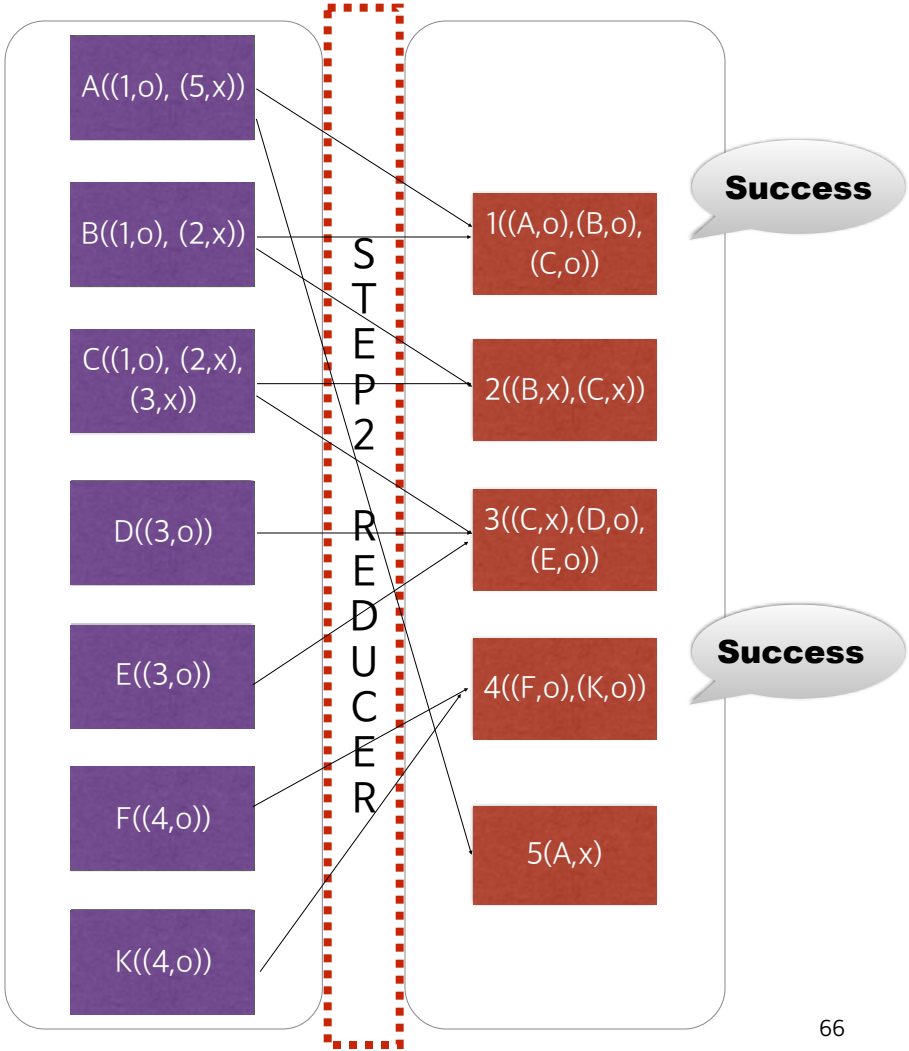


같은 Cell을 쓴 문서중
하나만 성공 후보로 선정
선정된 후보는 O
실패한 문서는 X

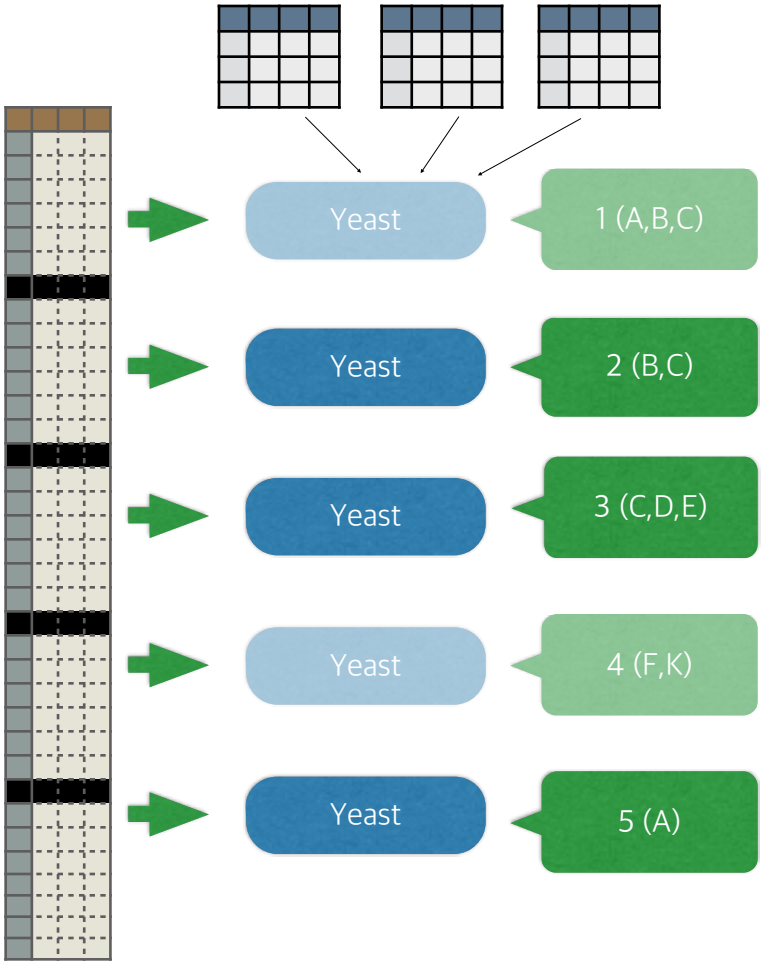

Yeast
1 (A,B,C)
Yeast
2 (B,C)
Yeast
3 (C,D,E)
Yeast
4 (F,K)
Yeast
5 (A)

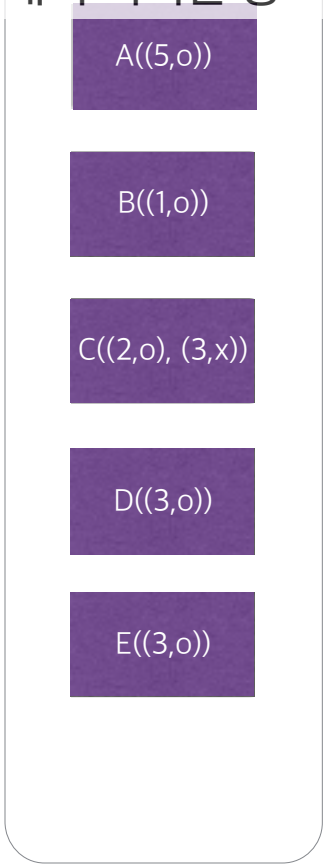

A((1,o), (5,x))
B((1,o), (2,x))
C((1,o), (2,x), (3,x))
D((3,o))
E((3,o))
F((4,o))
K((4,o))
STEP2 REDUCER
1((A,o),(B,o), (C,o))
2((B,x),(C,x))
3((C,x),(D,o), (E,o))
4((F,o),(K,o))
5(A,x)

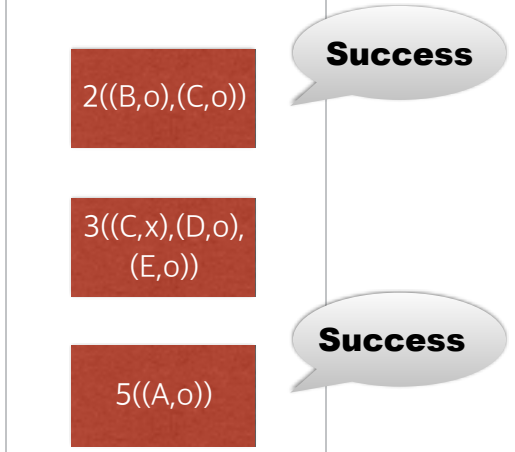

Yeast
1 (A,B,C)
Yeast
2 (B,C)
Yeast
3 (C,D,E)
Yeast
4 (F,K)
Yeast
5 (A)


A((1,o), (5,x))
B((1,o), (2,x))
C((1,o), (2,x), (3,x))
D((3,o))
E((3,o))
F((4,o))
K((4,o))
STEP2 REDUCER
1((A,o),(B,o), (C,o))
2((B,x),(C,x))
3((C,x),(D,o), (E,o))
4((F,o),(K,o))
5(A,x)
Success
Success

실패한 작업은 이 과정을 반복해서 수행하고,
실패 작업이 스트림 처리가 가능한 수준이면,
배치 처리는 종료







Yeast 배치 처리 성능 최적화

처리량을 늘리기 위해 Random Read를 최소화 해야 함

➡ 사용할 데이터를 scan해서 미리 메모리에 저장 (Prefetch)

스트림 처리에서의 Random Read


```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();
```

Yeast Layer

StorageSandbox

DDK Layer

Physical Storage

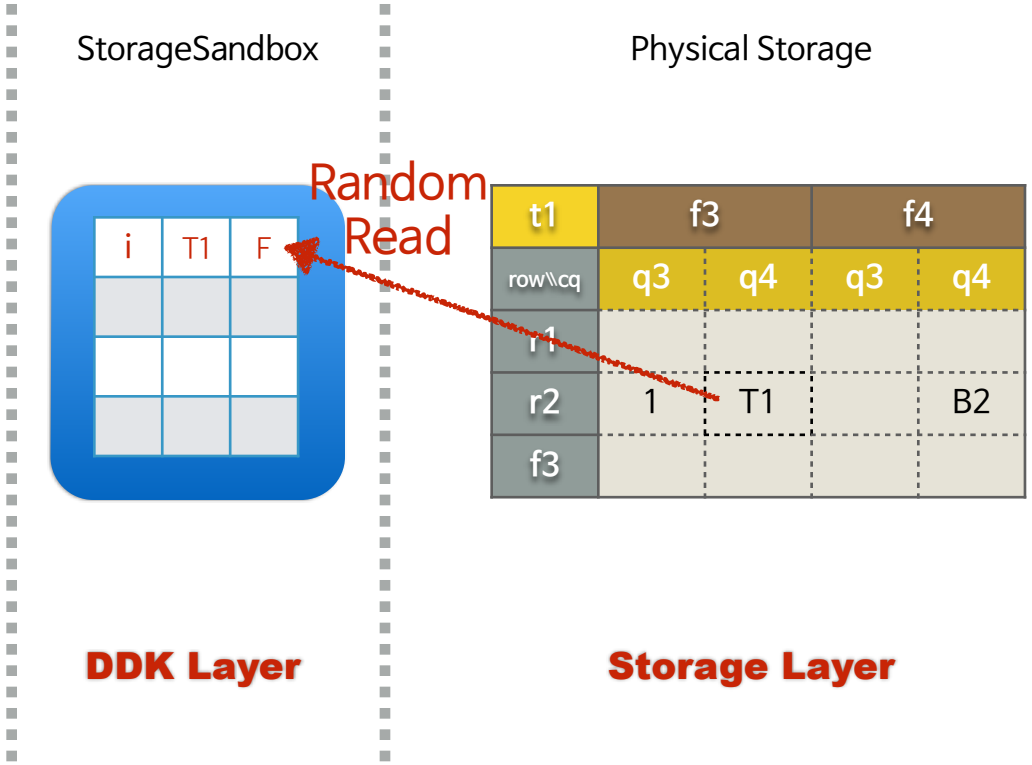
t1	f3		f4	
row\cq	q3	q4	q3	q4
r1				
r2	1	T1		B2
f3				

Storage Layer

스트림 처리에서의 Random Read


```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();
```

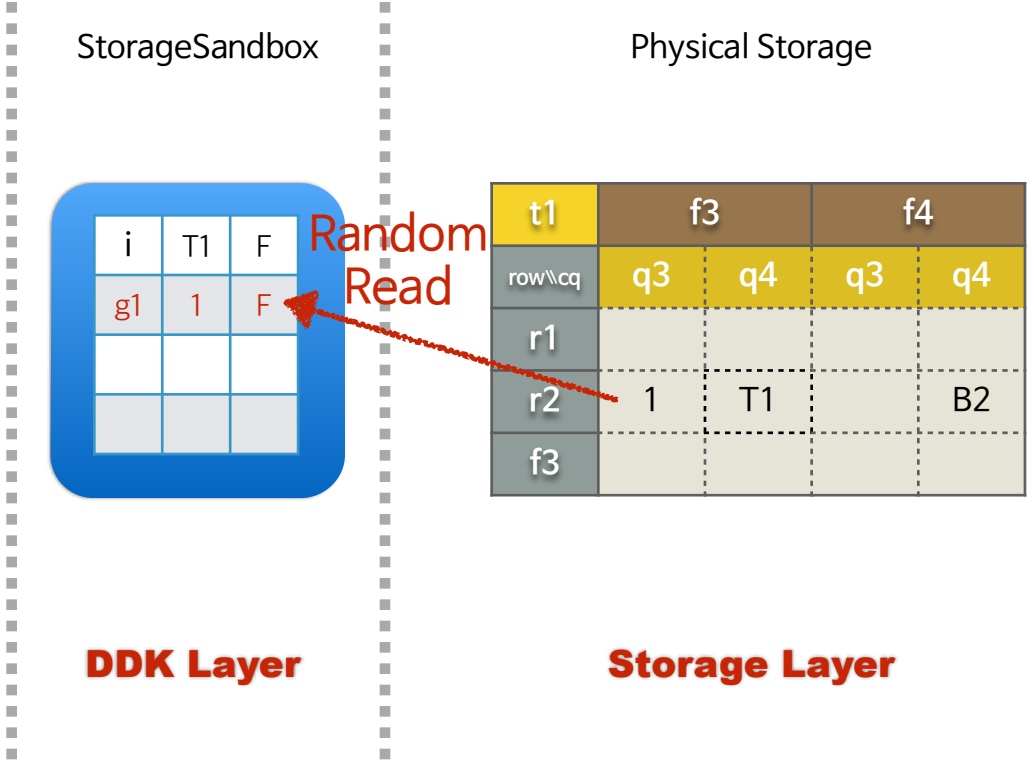
Yeast Layer



스트림 처리에서의 Random Read


```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();
```

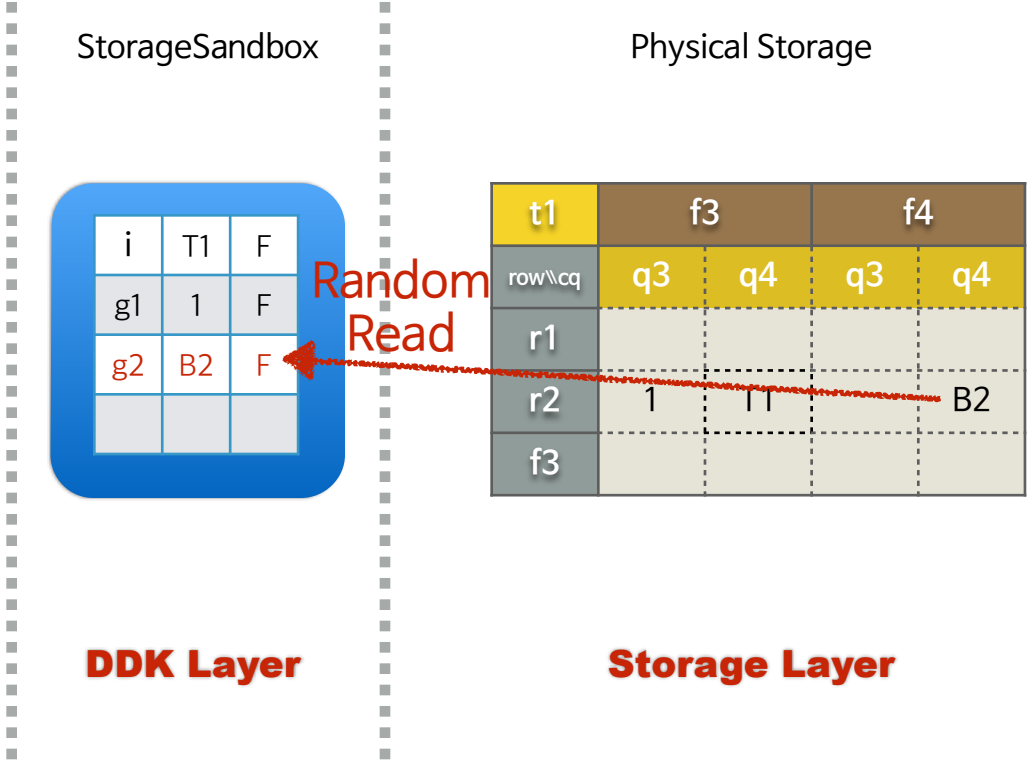
Yeast Layer



스트림 처리에서의 Random Read


```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();
```

Yeast Layer

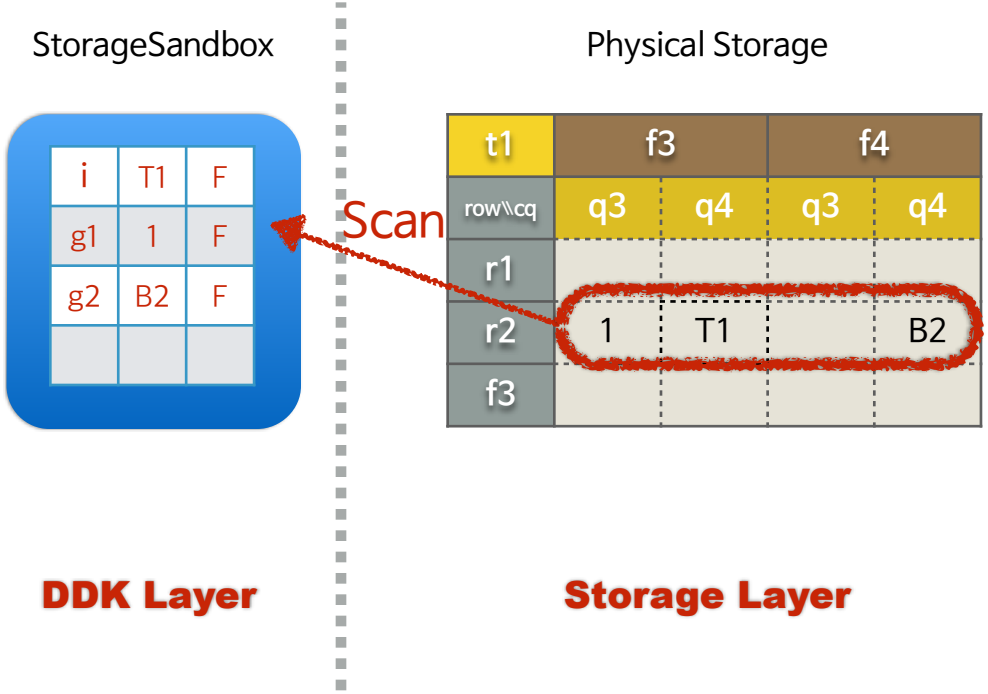


배치 처리의 Prefetch

Yeast 로직 수행 전에 데이터를 미리 메모리에 저장한다.

```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();
```

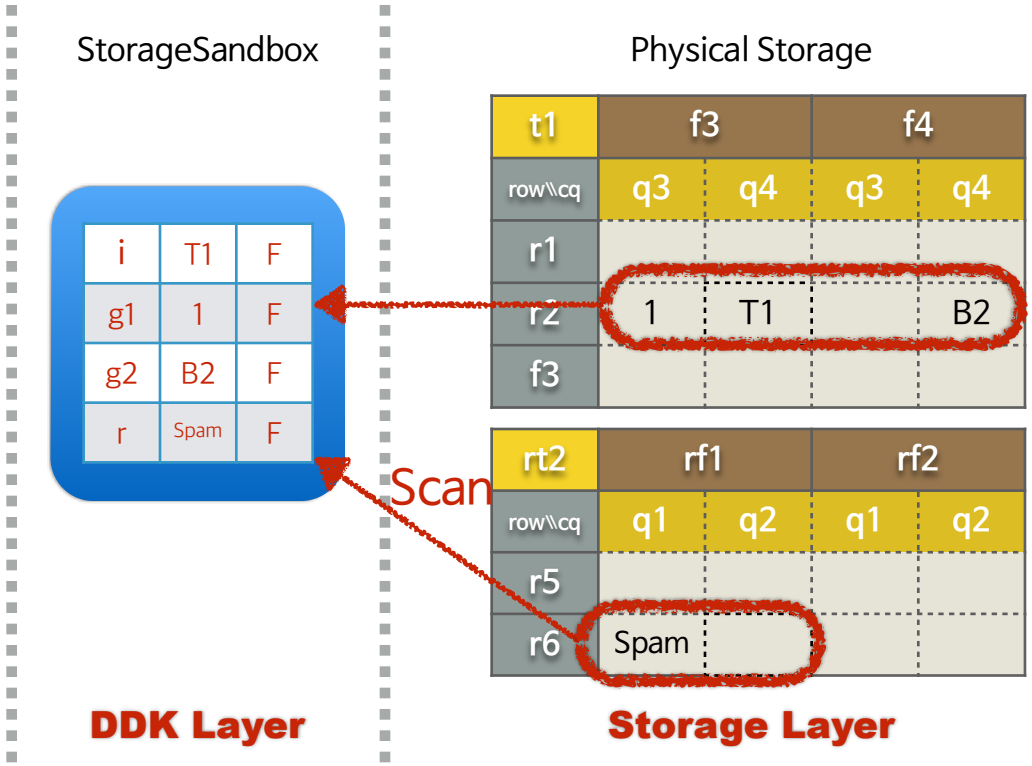
Yeast Layer



배치 처리의 Prefetch


```
Cell i = t1.r2.f3.q4
i.getString();

Cell g1 = t1.r2.f3.q3
g1.getInt();

Cell g2 = t1.r2.f4.q4
g2.getBytes();

Cell r = rt2.r6.rf1.q1
r.getString();
```

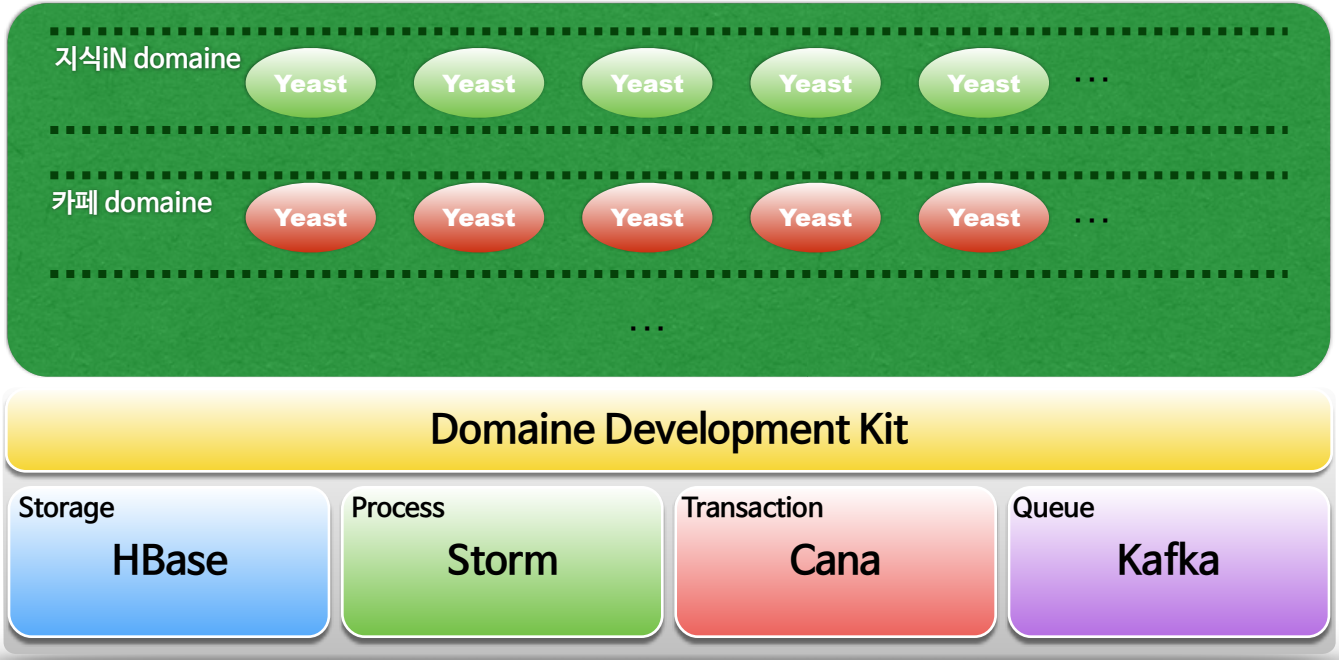
Yeast Layer



5.
Stack Overview

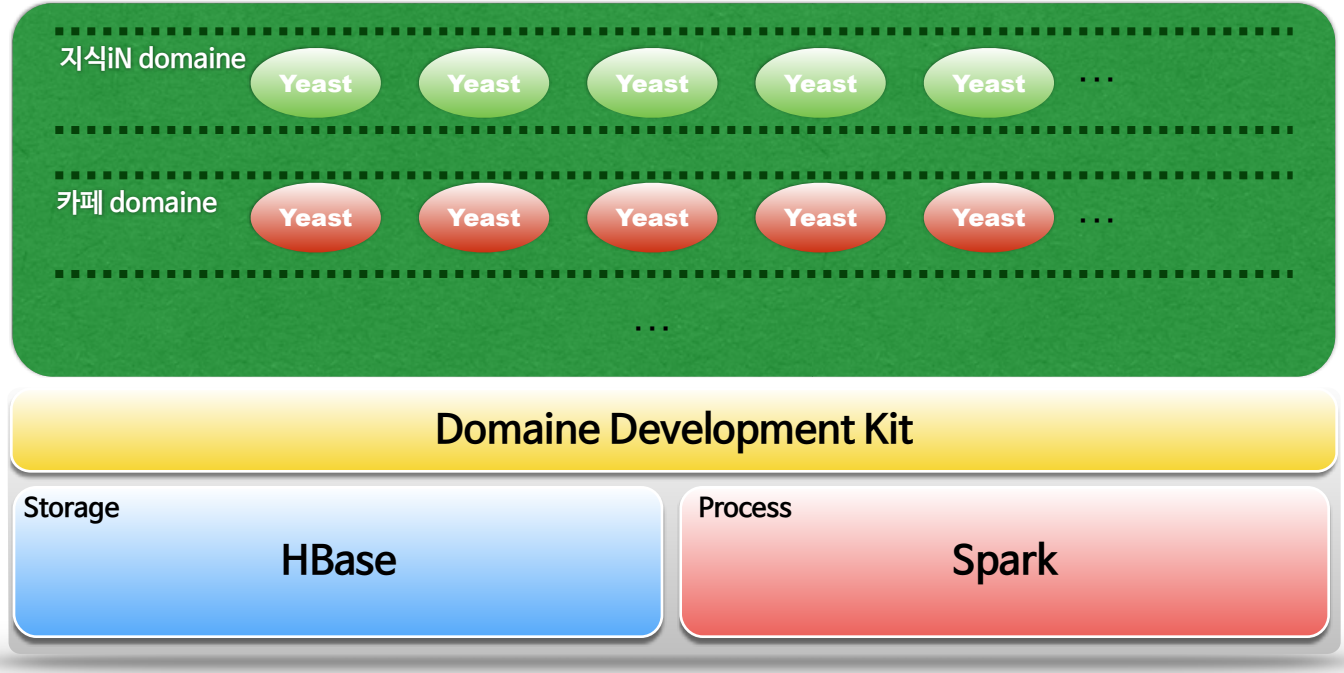
Domaine Architecture

Stack Overview – 스트림 처리



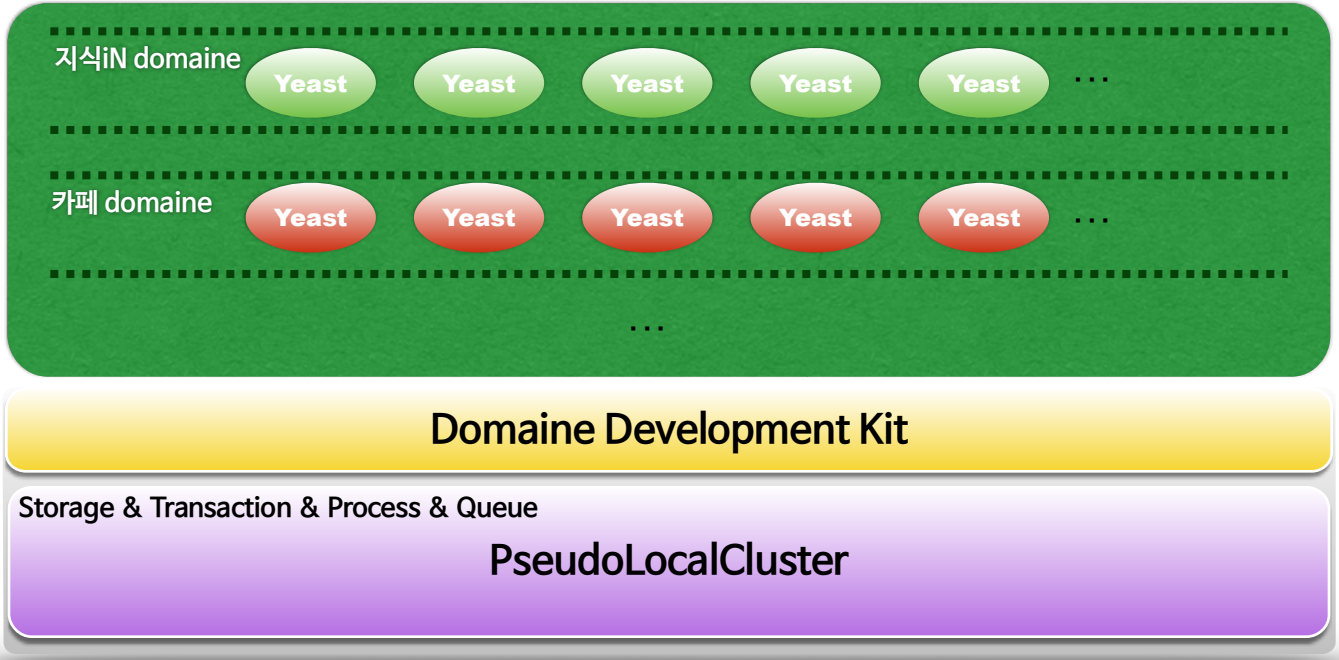
Domaine Architecture

Stack Overview – 배치 처리



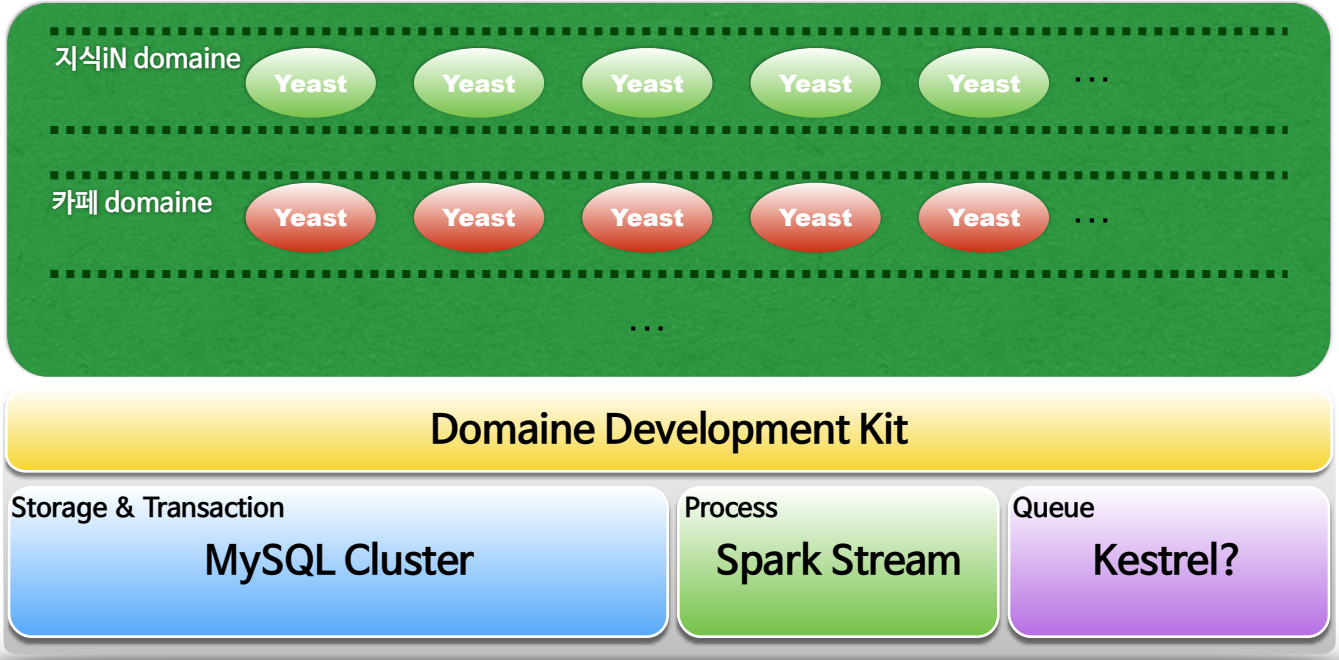
Domaine Architecture

Stack Overview – 디버깅 & 테스트



Domaine Architecture

Stack Overview – 대체가능 솔루션



6.
분산 환경 구성

분산 환경 구성

증분 시스템의 Storage 내용은 언제나 변하고 있음

➡ 주기적인 check point 관리가 중요

- Event queue offset => Kafka
- Storage snapshot => HBase

분산 환경 구성 – Check point

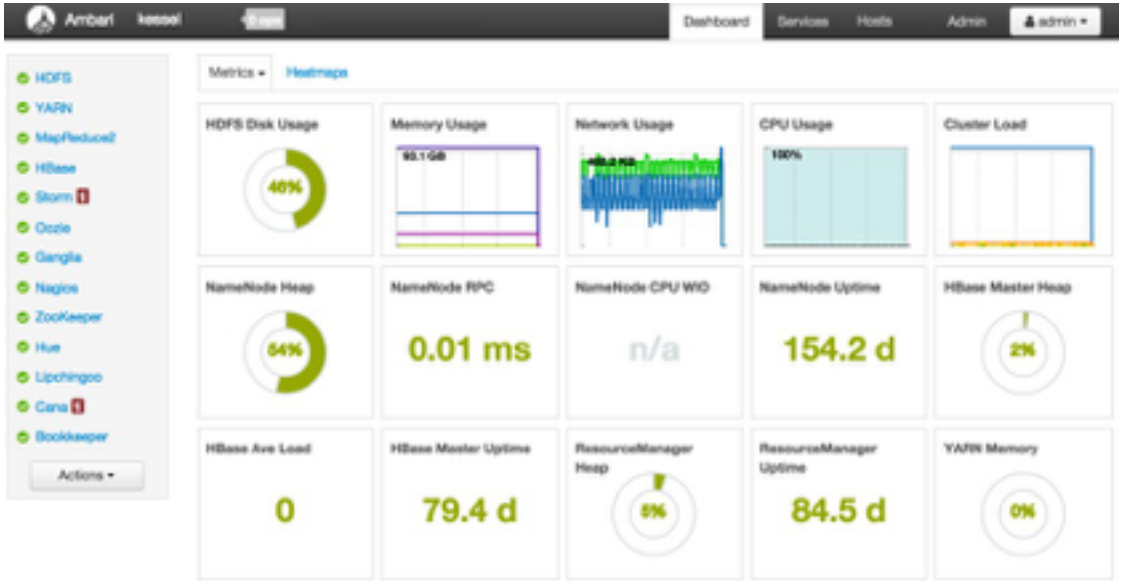
데이터 오염이 발생한 경우

- ➡ 오염 시점을 알면, Check point부터 스트림으로 재처리
- ➡ 시점을 알지 못하면, 초기 구축 배치 처리

분산 환경 구성

클러스터 관리 도구

- ➡ Ambari
- ➡ 필요 서비스 추가 및 stack 구성



분산 환경 구성

성능 지표 모니터링과 로그 검색

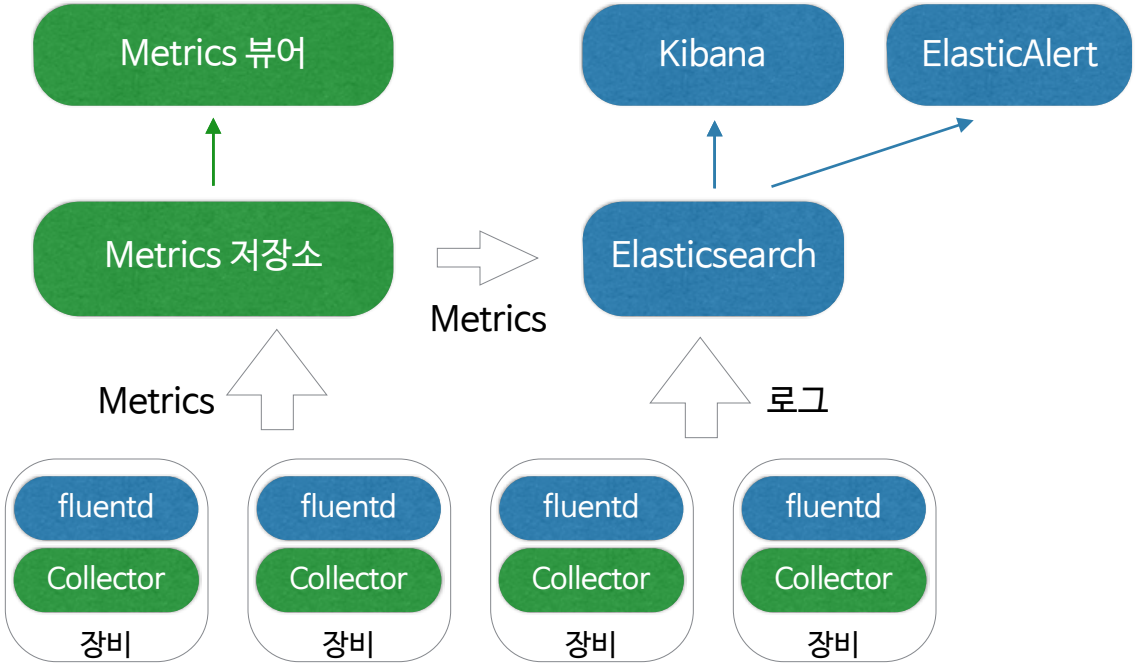
- ➡ 로그 검색 : 데이터 흐름 파악 용도
- ➡ 자체 모니터링 시스템 구축(Ganglia와 유사)
- ➡ Fluentd, ElasticSearch, Kibana

분산 환경 구성

시스템 상황에 대한 알람

- ➡ 알람을 위한 지표 저장 =〉 ElasticSearch
- ➡ 알람 전달 =〉ElastAlert

분산 환경 구성


Metrics 뷰어
Kibana
ElasticAlert
Metrics 저장소
Elasticsearch
Metrics
Metrics
로그
fluentd
Collector
장비
fluentd
Collector
장비
fluentd
Collector
장비
fluentd
Collector
장비

7.
적용 사례

증분 시스템 적용 후 변화

네이버 K 서비스		Timewise Staged 배치 시스템	증분 시스템
일주일 간 처리 문서 비율		100% (All Stages)	20.552%
갱신 시간	문서 갱신	최대 수 분 (Stage3)	최대 수 초 평균 55.198ms
	사용자 정보 갱신	최소 수분 (Stage3) ~ 최대 수일 (Stage1)	데이터 입수 즉시

마무리

증분 시스템이 만능은 아니다.

➡ 주어진 문제를 어떻게 해결하는 게 좋을지 고민이 필요

이제 시작일 뿐...

Q&A

Thank You