KOSSCON 2018 Tutorial

Beam Application을 Apache Nemo 를 활용하여 특정 환경에 최적화하여 실행하기

2018.11.29 서울대학교 SPL 송원욱, 서장호

Table of Contents

1. 사전 설치 및 준비
2. Apache Nemo 이해하기
3. Apache Beam 이해하기
4. Apache Nemo 위에서 Beam application 실행하기
5. 특정 환경에서 설정을 통해 바뀐 Beam application execution 관찰하기
6. 대규모 데이터 분석 시연 및 마무리

사전 설치 및 준비

실습 환경 갖추기 + 예제 코드 다운로드

다음 링크를 참고하여 실습 환경 설정 및 예제 코드 다운로드:

https://github.com/snuspl/kosscon

Java 및 Maven 설치 (Mac OS)

1. Homebrew가 없을 시, `$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"` 를 실행하여 Homebrew 설치
2. 다음 커맨드 차례대로 실행:

```
$ brew tap caskroom/versions
$ brew cask install java8
$ brew install maven
```

Java 및 Maven 설치 (Ubuntu)

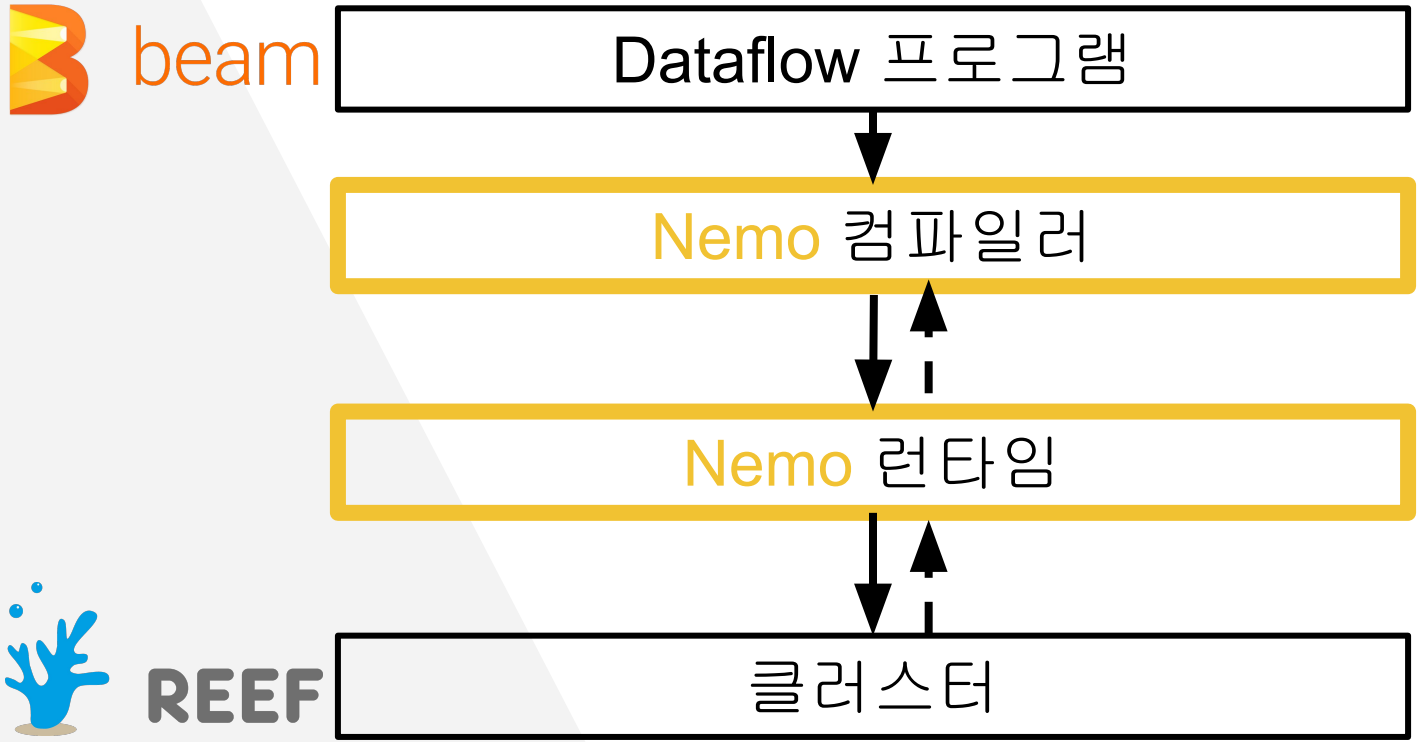
Ubuntu 16.04 이상의 버전을 권장.

1. `$ sudo apt update`
2. `$ sudo apt install openjdk-8-jdk maven`
3. `$ sudo update-java-alternatives /usr/lib/jvm/java-8-openjdk-amd64`

Apache Nemo 이해하기

Website: https://nemo.apache.org
GitHub: https://github.com/apache/incubator-nemo

Nemo 전체 그림



데이터 프로세싱 개요

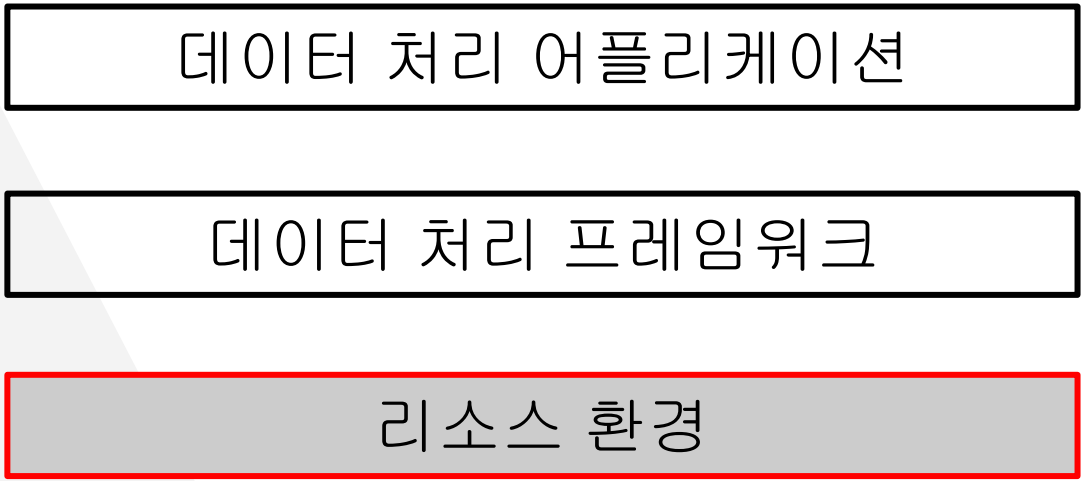
데이터 처리 어플리케이션

데이터 처리 프레임워크

리소스 환경

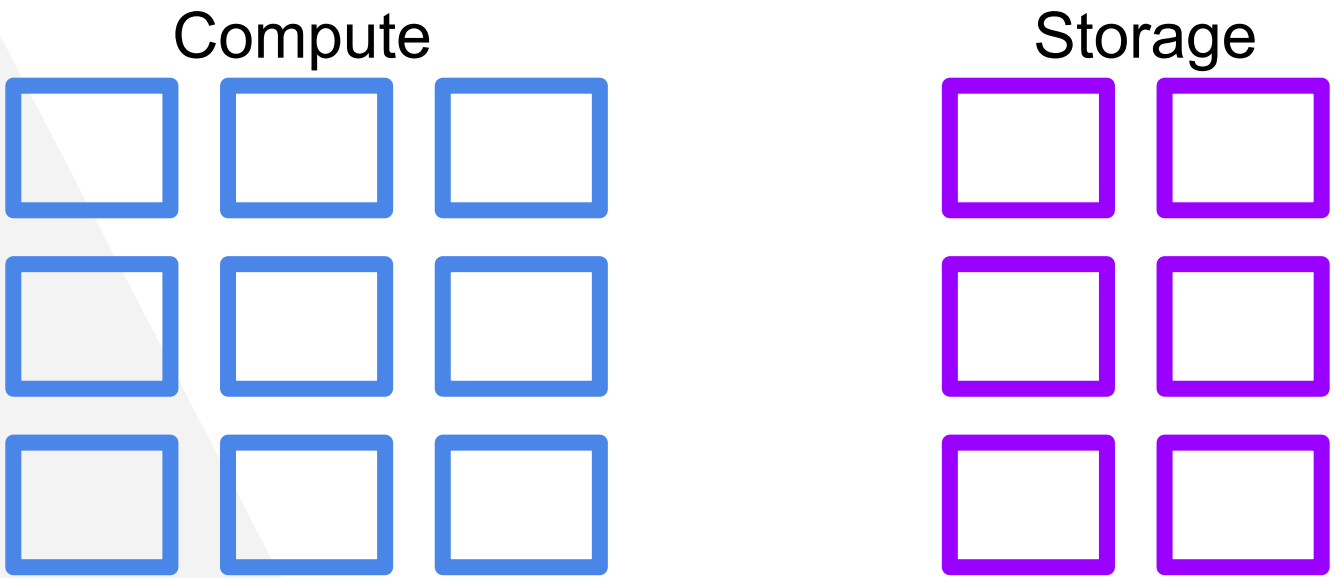
Spark, Flink,
Hadoop,
Dryad, Tez,
...

데이터 프로세싱 개요



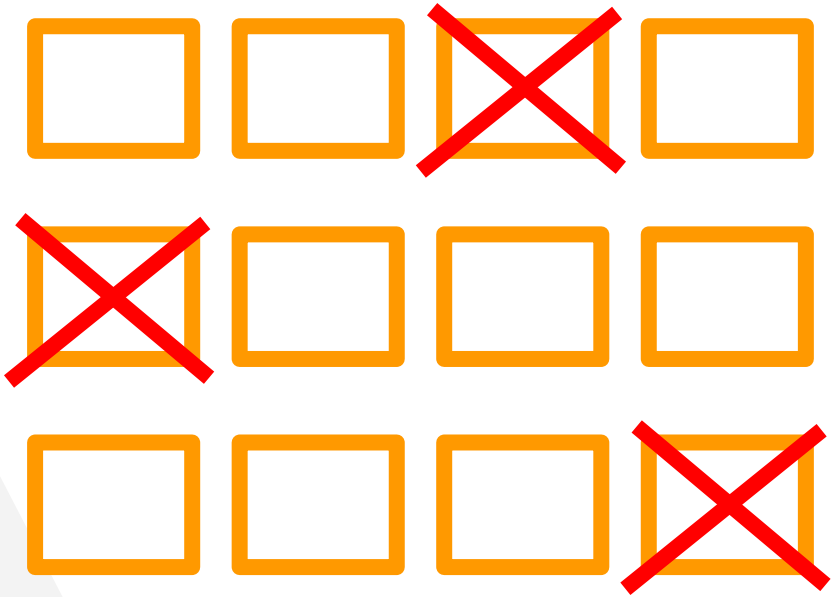
Spark, Flink,
Hadoop,
Dryad, Tez,
...

Disaggregation



Compute 노드에서 Storage 노드에
중간 데이터를 읽고 쓰는 환경

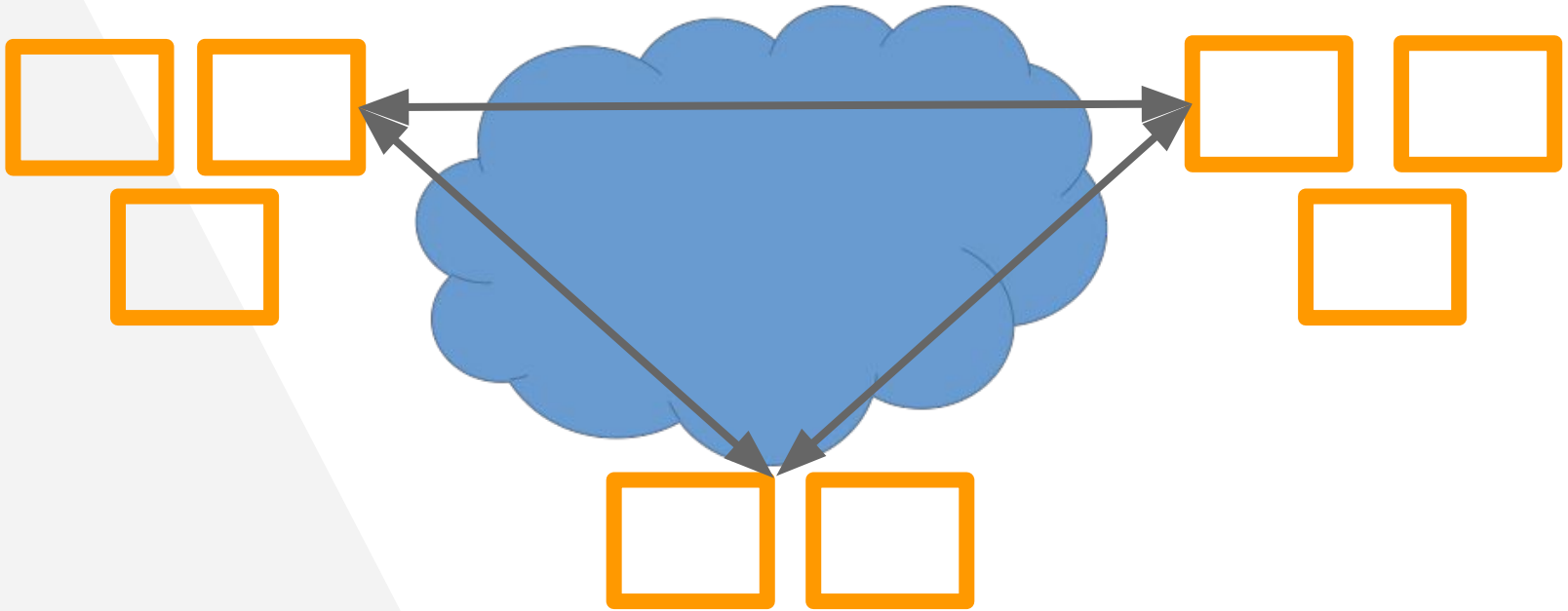
Transient Resources



Preemption!

Latency-critical job으로터 빌려온 자원은
예고 없이 회수될 수 있음

Cross Datacenter



WAN 대역폭은 좁고 비쌈

데이터 프로세싱 개요

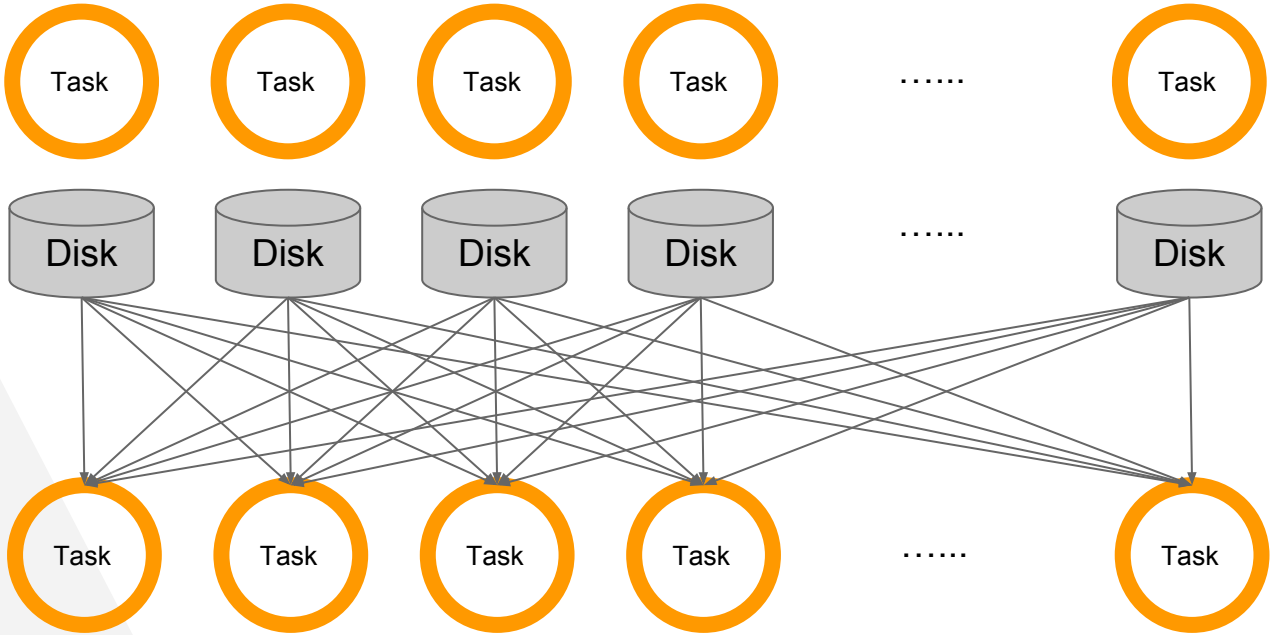
데이터 처리 어플리케이션

데이터 처리 프레임워크

리소스 환경

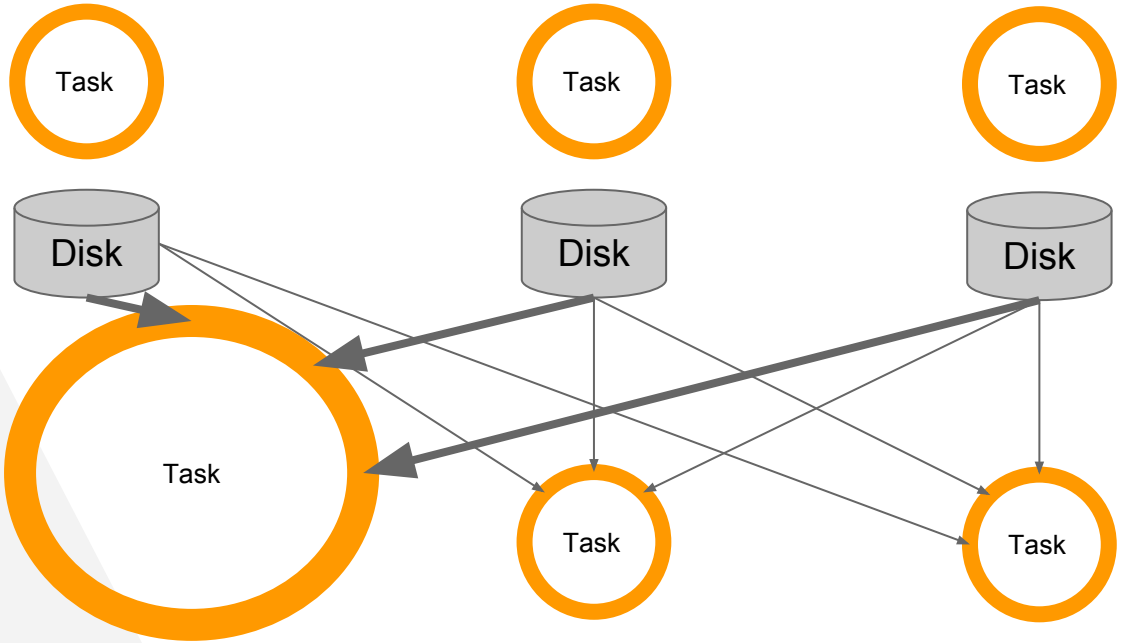
Spark, Flink, Hadoop, Dryad, Tez, ...

Large Shuffle



엄청난 양의 Random Disk Read 발생 → 느려짐

Skewed Data



특정 Popular한 Key 에 대해서 데이터 몰림 현상 발생

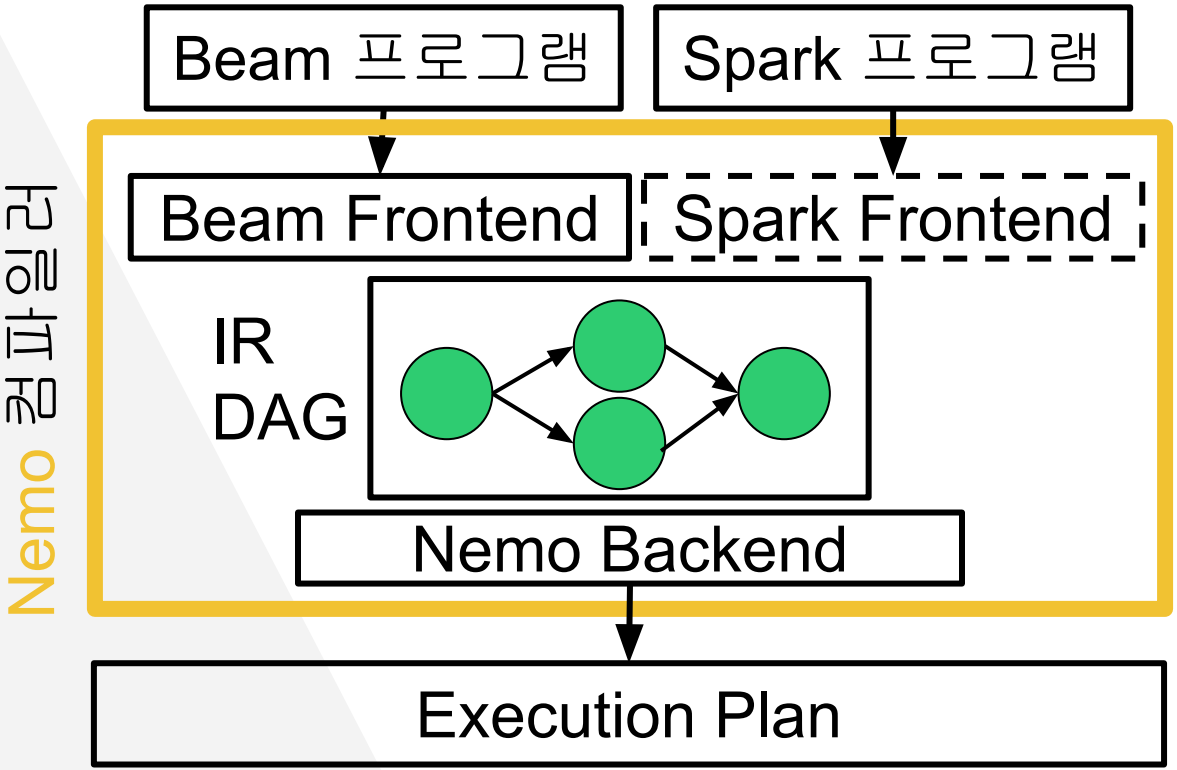
→ Straggler!

기존 시스템의 문제점

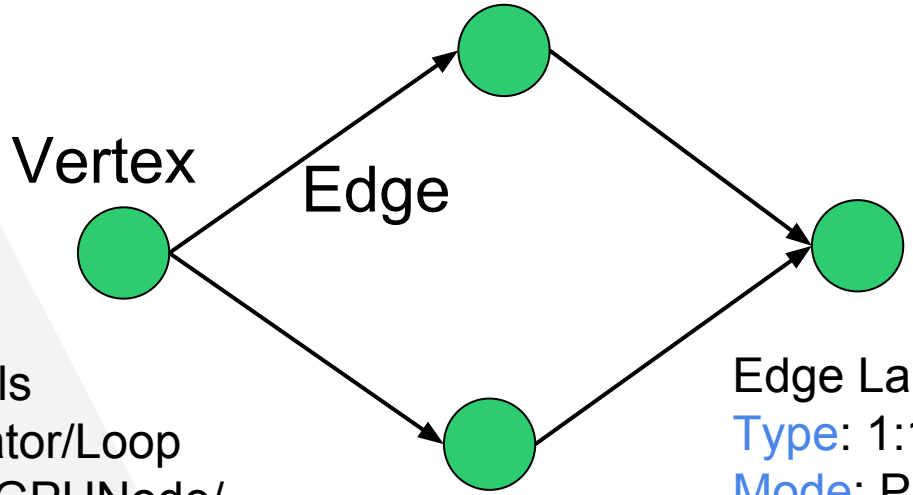
기존 시스템들: 한 가지의 설정 및 최적화만 적용 가능,
여러 설정들을 유연하게 적용하는 것 불가능

=> A new **flexible** and **extensible** data processing system

Nemo 컴파일러


Beam 프로그램
Spark 프로그램
Nemo 컴파일러
Beam Frontend
Spark Frontend
IR
DAG
Nemo Backend
Execution Plan

IR (Intermediate Representation) DAG
Annotation을 할 수 있는, Language에 얽매이지 않는 DAG

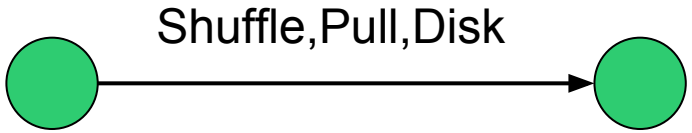


Vertex Labels
Type: Operator/Loop
Placement: GPUNode/
ReservedNode/TransientNode/Any
Parallelism

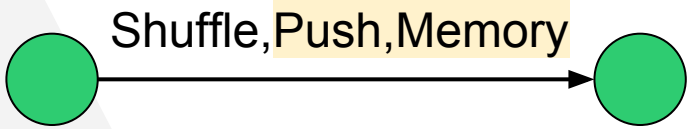
Edge Labels
Type: 1:1/Broadcast/Shuffle
Mode: Push/Pull
Storage: Memory/Disk/RemoteDisk

MapReduce IR DAG 예시

Classical MapReduce



Small-scale MapReduce



컴파일러에서 런타임으로

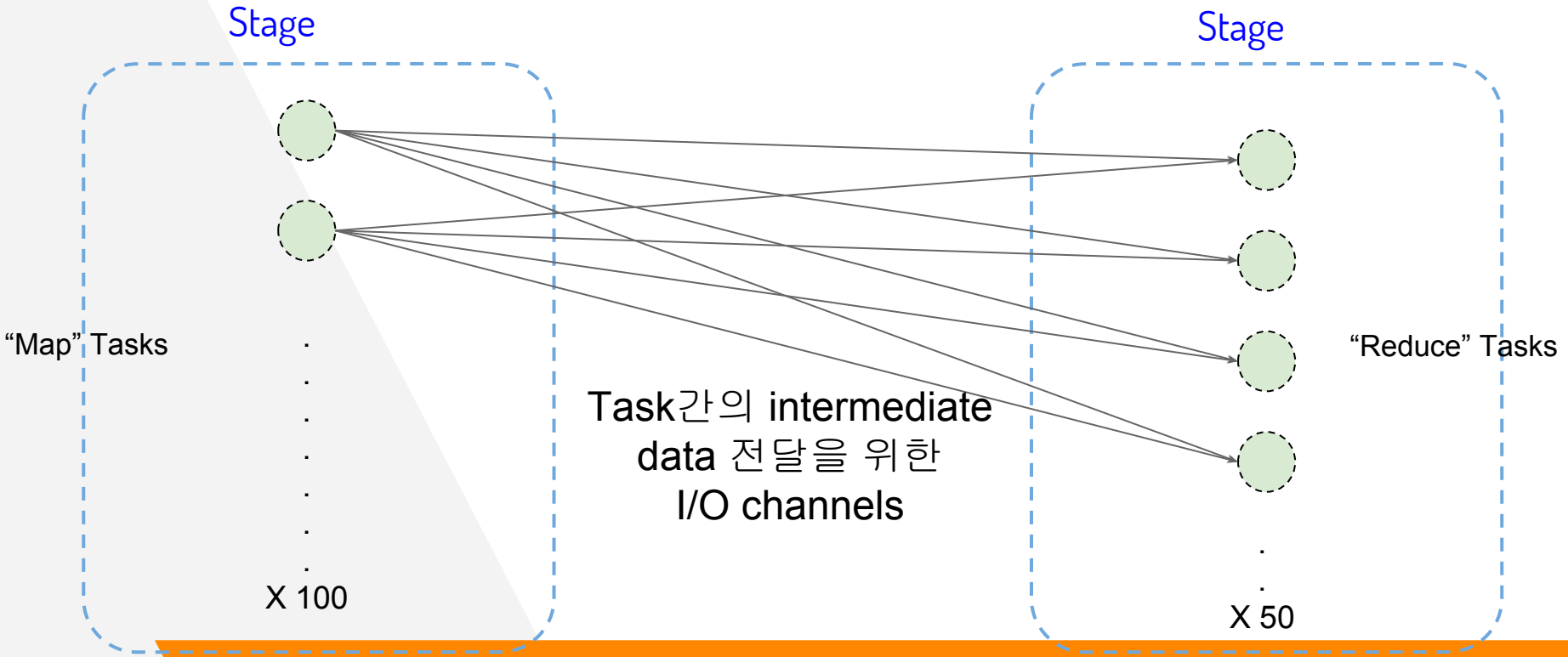
최적화 된 IR DAG

Type: “Map” Operator
Placement: “Reserved” Node
Parallelism: 100

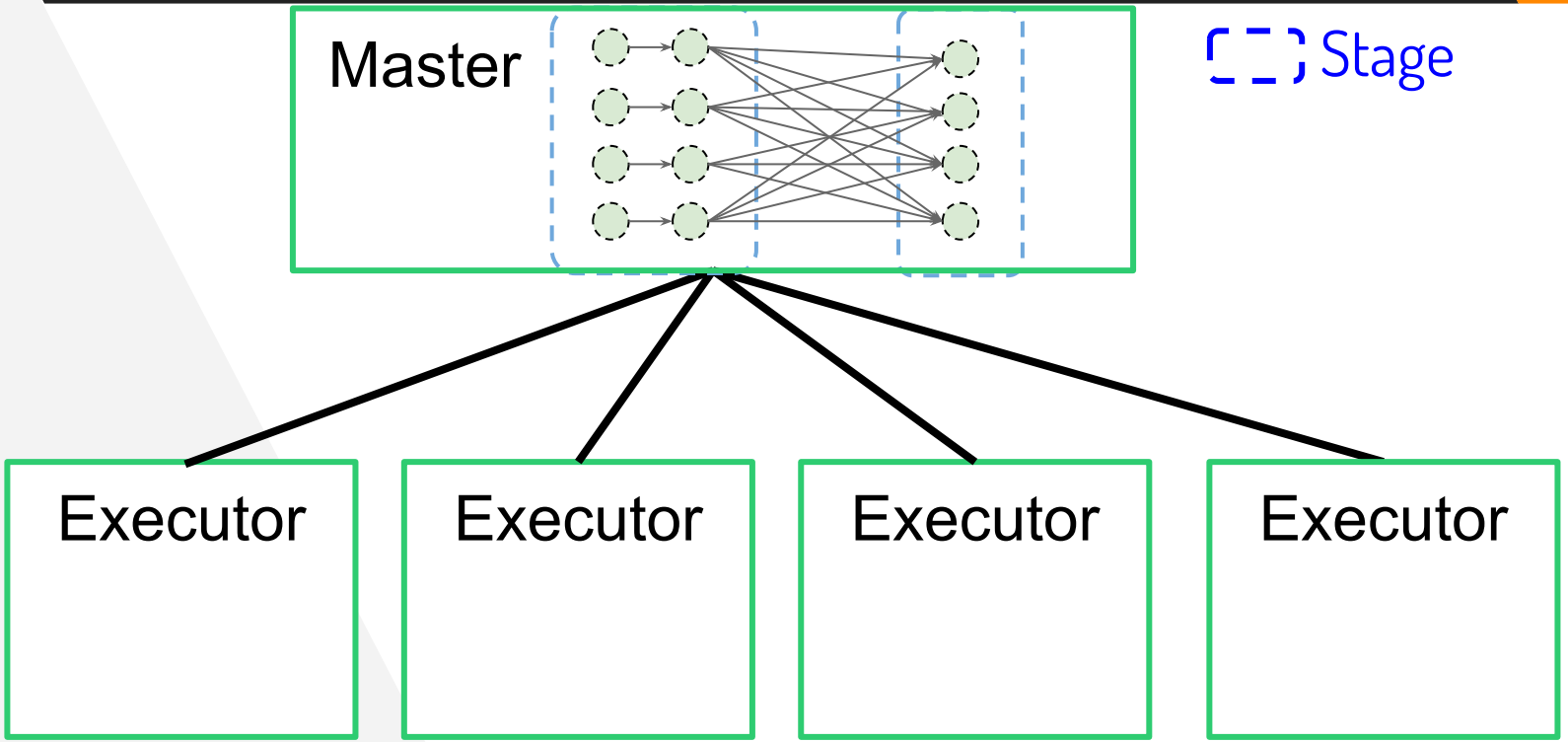
Type: “Reduce” Operator
Placement: “Reserved” Node
Parallelism: 50

컴파일러에서 런타임으로

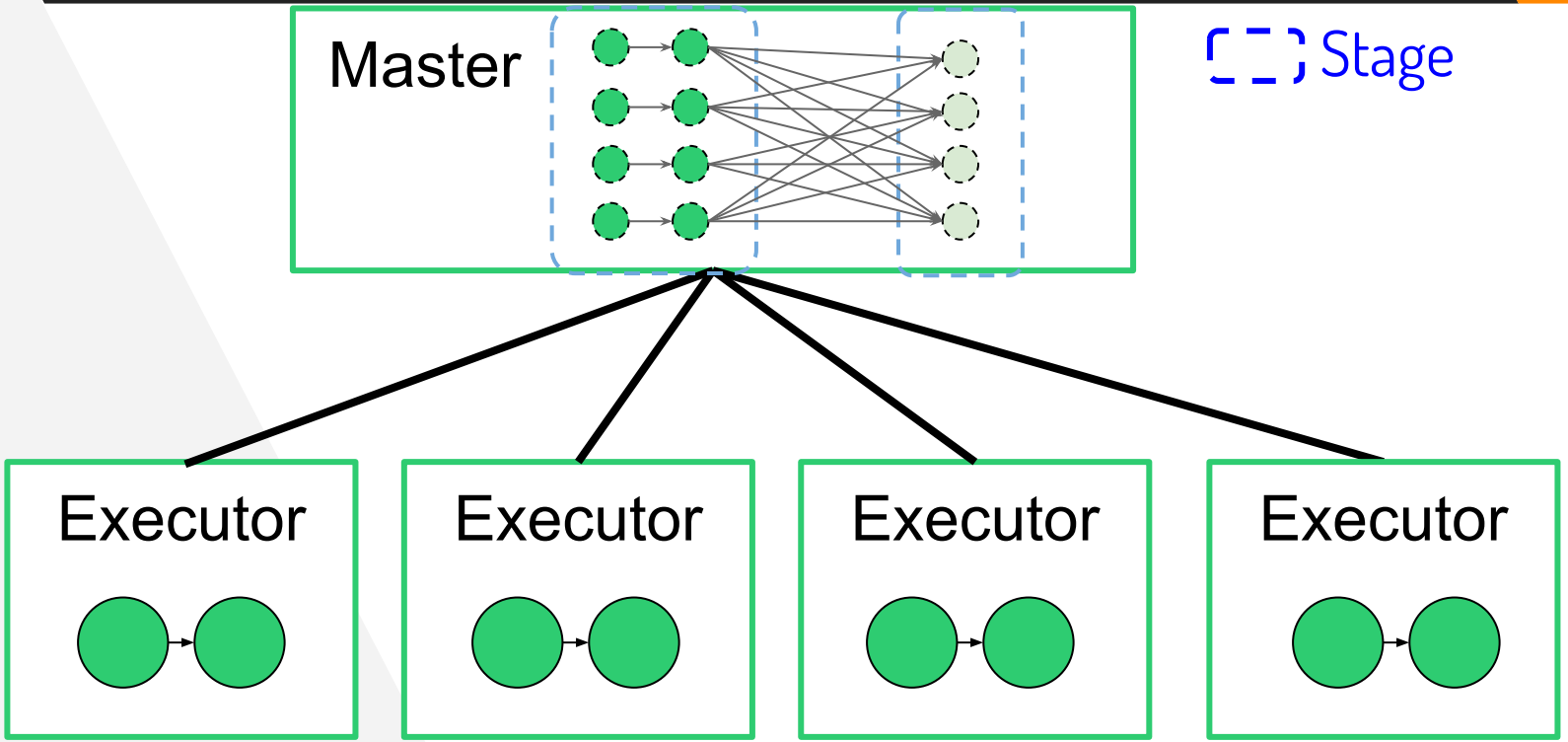
Execution Plan



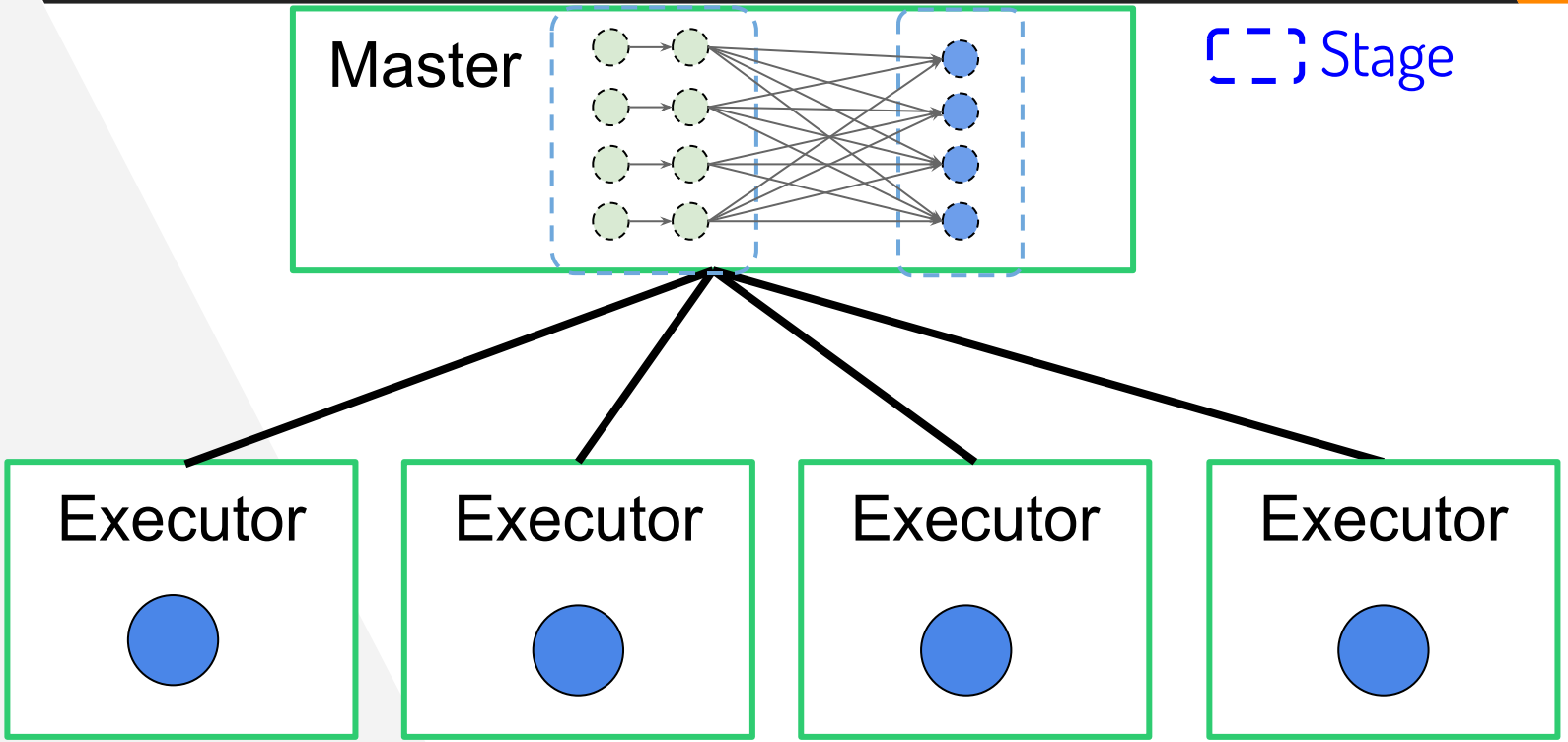
Nemo 런타임에서의 분산 Execution



Nemo 런타임에서의 분산 Execution



Nemo 런타임에서의 분산 Execution

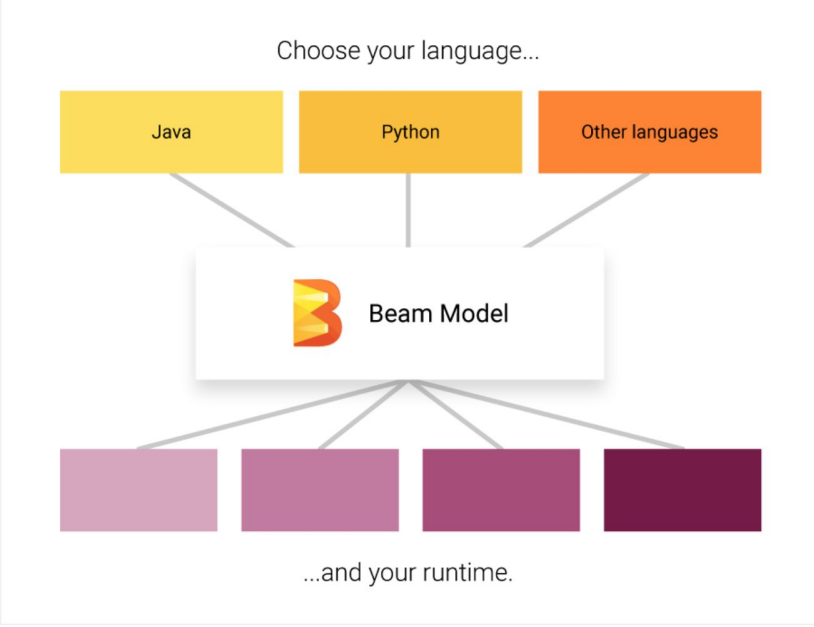


Beam Application 이해하기

Website: https://beam.apache.org
GitHub: https://github.com/apache/beam

Apache Beam?

- MapReduce, FlumeJava, Spark 등
- 통합된 프로그래밍 모델
- Batch + Stream
- Nemo를 비롯한 여러 runtime 지원
  - (e.g., Spark, Flink)



Beam Application의 예시: WordCount

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6         .apply(
7             FlatMapElements.into(TypeDescriptors.strings())
8                 .via((String line) -> Arrays.asList(line.split("[^\\p{L}]+"))))
9         .apply(Filter.by((String word) -> !word.isEmpty()))
10        .apply(Count.perElement())
11        .apply(
12            MapElements.into(TypeDescriptors.strings())
13                .via(
14                    (KV<String, Long> wordCount) ->
15                        wordCount.getKey() + ": " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

Beam Application의 예시 Walkthrough: Creating the Pipeline

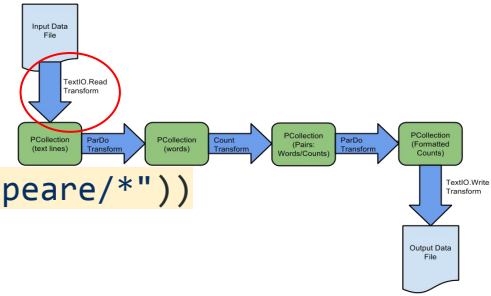
```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-sa
6         .apply(
7             FlatMapElements.into(TypeDescriptors.st
8                 .via((String line) -> Arrays.asList
9         .apply(Filter.by((String word) -> !word.isE
10        .apply(Count.perElement())
11
12                                                    ))
13
14
15                        wordCount.getKey() + ": " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

PipelineOptions를 통해 Pipeline에 대해 여러가지 설정을 세팅해 줄 수 있다 (e.g., pipeline runner: NemoRunner / SparkRunner). 이 예시에서는 기본값인 DirectRunner을 사용하게 된다.

위에서 세팅한 option을 가지고, Pipeline을 만든다. 이 Pipeline object가 여러 transformation으로 이루어진 그래프를 만들게 된다.

Beam Application의 예시 Walkthrough: Reading input (from text files)

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6         .apply(
7             FlatMapElements.into(TypeDescriptors.strings())
8                 .via((String line) -> Arrays.asList(line.split("[^\\p{L}]+"))))
9         .apply(Filter.by((String word) -> !word.isEmpty()))
10
11
12                                                          ings())
13
14                                                           ->
15                                                          " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

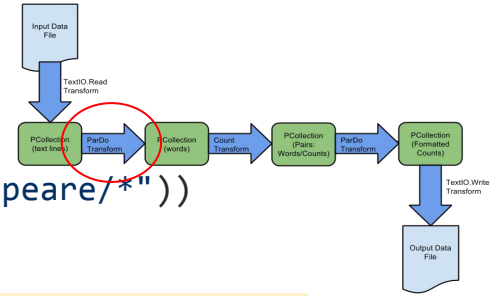


TextIO.Read transform을 적용한다. 이건 맨 처음의 PCollection object를 만든다. 이 상태에서의 PCollection은 텍스트 파일들의 각 line들이다. gs는 public하게 접근 가능한 google cloud storage bucket을 나타낸다.

Beam Application의 예시 Walkthrough: Applying ParDo transforms

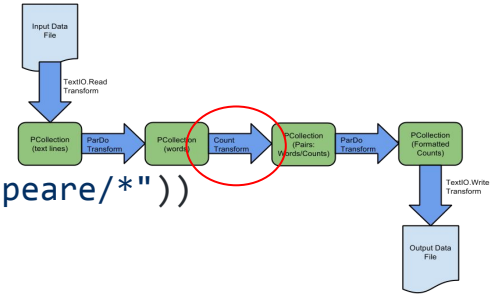
```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6         .apply(
7             FlatMapElements.into(TypeDescriptors.strings())
8                 .via((String line) -> Arrays.asList(line.split("[^\\p{L}]+"))))
9         .apply(Filter.by((String word) -> !word.isEmpty()))
10        .apply(Count.perElement())
11
12                                                              ))
13
14
15                                                      ordCount.getValue()))
16
17    p.run();
18  }
19 }
```

위의 ParDo (Parallel-Do) transform은 각 line에서 word들을 뽑아내서 PCollection을 word들의 List로 만든다. 주목할 점은 여기에서 하나하나의 line에 대해서 똑같은 DoFn을 적용한다는 점이다. (DoFn 은 in-line으로 anonymous class로 정의됨)



Beam Application의 예시 Walkthrough: Applying SDK-provided transforms (Count)

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6         .apply(
7             FlatMapElements.into(TypeDescriptors.strings())
8                 .via((String line) -> Arrays.asList(line.split("[^\\p{L}]+"))))
9         .apply(Filter.by((String word) -> !word.isEmpty()))
10        .apply(Count.perElement())
11        .apply(
12            Ma
13
14
15                                                    nt.getValue()))
16        .apply(
17    p.run();
18  }
19 }
```

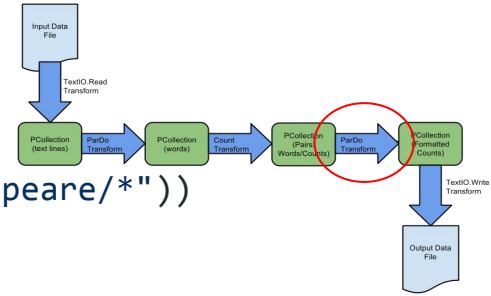


SDK에서는 여러가지 transform들을 기본적으로 제공하는데, 일단 비어있는 word들을 Filter을 통해 제외시키고, Count를 통해 주어진 PCollection에 있는 각 word들의 개수를 key/value pair으로 나타낸다.

Beam Application의 예시 Walkthrough: Applying ParDo transforms

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3                                                    tory.create();
4
5                                                   -samples/shakespeare/*"))
6
7                                                   .strings())
8                                                   ist(line.split("[^\\p{L}]+"))))
9         .apply(Filter.by((String word) -> !word.isEmpty()))
10        .apply(Count.perElement())
11        .apply(
12            MapElements.into(TypeDescriptors.strings())
13                .via(
14                    (KV<String, Long> wordCount) ->
15                        wordCount.getKey() + ": " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

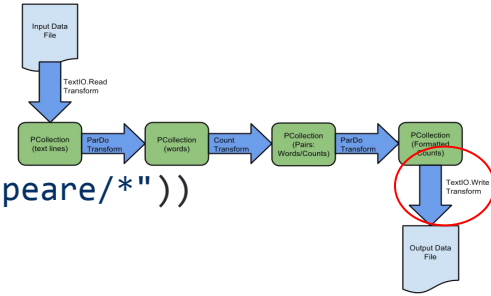
다음 ParDo transform에서는 앞에서 Count를 통해서 key/value pair으로 저장된 PCollection의 각 항목을 출력할 수 있는 string으로 만드는 작업을 한다. 아래 코드에서는 각 항목을 key + “: ” + value라는 스트링으로 만든다.



Beam Application의 예시 Walkthrough: Writing output (to a text file)

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6     .apply(
7  rings())
8  (word.split("[^\\p{L}]+"))))
9  mpty()))
10
11        .apply(
12            MapElements.into(TypeDescriptors.strings())
13                .via(
14                    (KV<String, Long> wordCount) ->
15                        wordCount.getKey() + ": " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

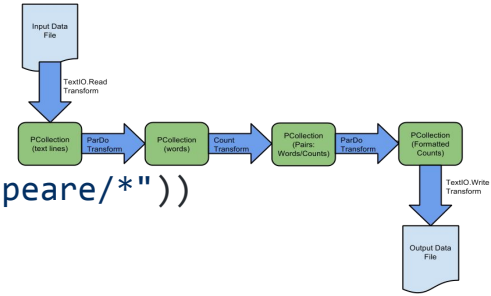
이제 "word: 3" 식으로 저장되어있는 string들의 PCollection을 wordcounts라는 이름의 output text file에 쓰게 된다. 각 string이 output file의 line이 된다.



Beam Application의 예시 Walkthrough: Running the Pipeline

```
1  public class MinimalWordCount {
2   public static void main(String[] args) {
3     PipelineOptions options = PipelineOptionsFactory.create();
4     Pipeline p = Pipeline.create(options);
5     p.apply(TextIO.read().from("gs://apache-beam-samples/shakespeare/*"))
6         .apply(
7                                                    s.strings())
8                                                    List(word.split("[^\\p{L}]+"))))
9                                                    .isEmpty()))
10
11        .apply(
12            MapElements.into(TypeDescriptors.strings())
13                .via(
14                    (KV<String, Long> wordCount) ->
15                        wordCount.getKey() + ": " + wordCount.getValue()))
16        .apply(TextIO.write().to("wordcounts"));
17    p.run();
18  }
19 }
```

이제 마지막으로, 주어진 option과 앞에서 정의한 transform들의 그래프로 이루어진 Pipeline을 실제로 실행한다.



Apache Nemo 위에서 Beam Application 실행하기

MapReduce Beam Application 실행하기

1. 앞서 다운받은 압축파일에서 압축 해제
2. 다음 코드 실행

```
./bin/run_beam.sh \
    -job_id mr_default \
    -user_main org.apache.nemo.examples.beam.MinimalWordCount \
    -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount"
```

3. 결과를 Web-UI를 사용해서 관찰

MapReduce Beam Application 실행하기

1. 앞서 다운받은 압축파일에서 압축 해제
2. 다음 코드 실행

Job ID를 'mr_default'로 설정

```
./bin/run_beam.sh \
    -job_id mr_default \
    -user_main org.apache.nemo.examples.beam.MinimalWordCount \
    -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount"
```

3. 결과를 Web-UI를 사용해서 관찰

MapReduce Beam Application 실행하기

1. 앞서 다운받은 압축파일에서 압축 해제
2. 다음 코드

아까전에 살펴 본 MinimalWordCount
프로그램을 실행할 것이라고 말해주기

```
./bin/run_beam.sh \
    -job_id mr_default \
    -user_main org.apache.nemo.examples.beam.MinimalWordCount \
    -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount"
```

3. 결과를 Web-UI를 사용해서 관찰

MapReduce Beam Application 실행하기

1. 앞서 다운받은 압축파일에서 압축 해제
2. 다음 코드

WordCount에서 사용할 input과 결과를 출력할 output 파일 지정해주기

```
./bin/run_beam.sh \
    -job_id mr_default \
    -user_main org.apache.nemo.examples.beam.MinimalWordCount \
    -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount"
```

3. 결과를 Web-UI를 사용해서 관찰

MapReduce Beam Application 실행하기

1. 앞서 다운받은 압축파일에서 압축 해제
2. 다음 코드 실행
3. 결과를 Web-UI를 사용해서 관찰

Web UI 직접 보기

Alternating Least Squares Beam Application 실행하기

1. 다음 코드 실행

```
./bin/run_beam.sh \
    -job_id als \
    -user_main org.apache.nemo.examples.beam.AlternatingLeastSquare \
    -user_args "`pwd`/test_input_als 10 3"
```

2. 결과를 Web-UI를 사용해서 관찰

Alternating Least Squares Beam Application 실행하기

1. 다음 코드 실행

Job ID를 'als'로 설정

AlternatingLeastSquare 프로그램을 실행할 것이라고 말해주기

```
./bin/run_beam.sh \
    -job_id als \
    -user_main org.apache.nemo.examples.beam.AlternatingLeastSquare \
    -user_args "`pwd`/test_input_als 10 3"
```

2. 결과를 Web-UI를 사용해서 관찰

WordCount에서 사용할 input과 결과를 출력할 output 파일 지정해주기

Alternating Least Squares Beam Application 실행하기

1. 다음 코드 실행
2. 결과를 Web-UI를 사용해서 관찰

Web UI 직접 보기 (2)

설정을 통해 바뀐 Beam Application 실행 및 관찰하기

APIs

CompileTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t);`
RunTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t, T t);`
PolicyBuilder	`PolicyBuilder registerCompileTimePass(CompileTimePass compileTimePass);`
PolicyBuilder	`PolicyBuilder registerRuntimePass(RuntimePass<?> runtimePass, CompileTimePass runtimePassRegistrar);`
PolicyBuilder	`Policy build();`
IRVertex/IREdge	`IRVertex/IREdge setProperty(ExecutionProperty<?> executionProperty);`
IRVertex/IREdge	`Optional<T> getPropertyValue(Class<? extends ExecutionProperty<T>> executionPropertyKey);`

APIs

IR DAG → IR DAG function

(IR DAG, T) → IR DAG function

CompileTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t);`
RunTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t, T t);`
PolicyBuilder	`PolicyBuilder registerCompileTimePass(CompileTimePass compileTimePass);`
PolicyBuilder	`PolicyBuilder registerRuntimePass(RuntimePass<?> runtimePass, CompileTimePass runtimePassRegistrar);`
PolicyBuilder	`Policy build();`
IRVertex/IREdge	`IRVertex/IREdge setProperty(ExecutionProperty<?> executionProperty);`
IRVertex/IREdge	`Optional<T> getPropertyValue(Class<? extends ExecutionProperty<T>> executionPropertyKey);`

APIs

CompileTimePass	apply(DAG<IRVer
RunTimePass	DAG<IRVertex, IREdge> apply(DAG<IRVer
PolicyBuilder	PolicyBuilder registerCompileTimePass(CompileTimePass *compileTimePass*);
PolicyBuilder	PolicyBuilder registerRuntimePass(RuntimePass<?> *runtimePass*, CompileTimePass *runtimePassRegistrar*);
PolicyBuilder	Policy build();
IRVertex/IREdge	IRVertex/IREdge setProperty(ExecutionProperty<?> *executionProperty*);
IRVertex/IREdge	Optional<T> getPropertyValue(tionProperty<T>> *executionPropertyKey*);

Compile-Time에 일어나는 Pass를 등록

Run-time에 일어나는 Pass를 등록. 이 때, Run-time pass를 trigger 하는 Compile-time pass를 같이 등록해야 함

Policy 만들기

APIs

CompileTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t);`
RunTimePass	`DAG<IRVertex, IREdge> apply(DAG<IRVertex, IREdge> t, T t);`
PolicyBuilder	`PolicyBuilder registerCompileTimePass(CompileTimePass compileTimePass);`
PolicyBuilder	`PolicyBuilder registerRuntime` … `ePassRe`
PolicyBuilder	`Policy build();`
IRVertex/IREdge	`IRVertex/IREdge setProperty(ExecutionProperty<?> executionProperty);`
IRVertex/IREdge	`Optional<T> getPropertyValue(Class<? extends ExecutionProperty<T>> executionPropertyKey);`

Push/Pull등 Property 정해주기

정해진 Property 불러오기

Pado 설정을 적용한 예시

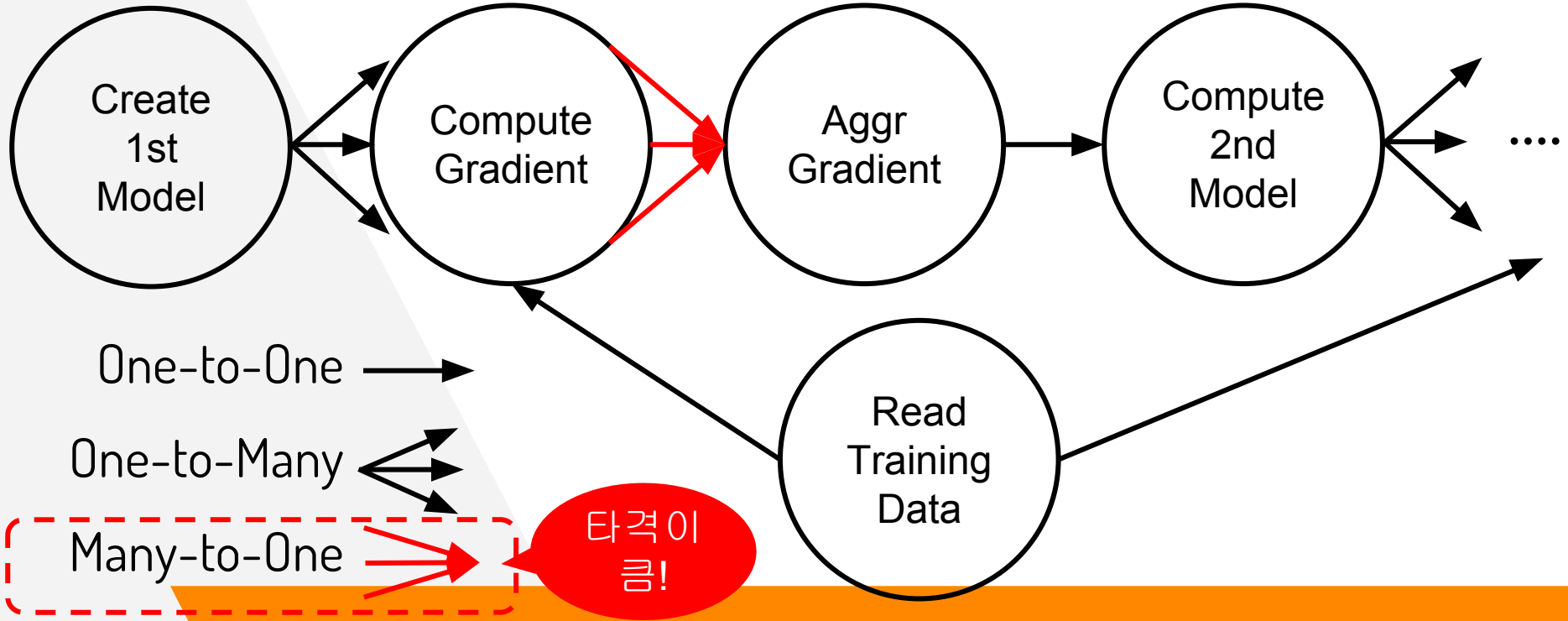
1. 다음 코드 실행

TransientResourcePolicy 적용

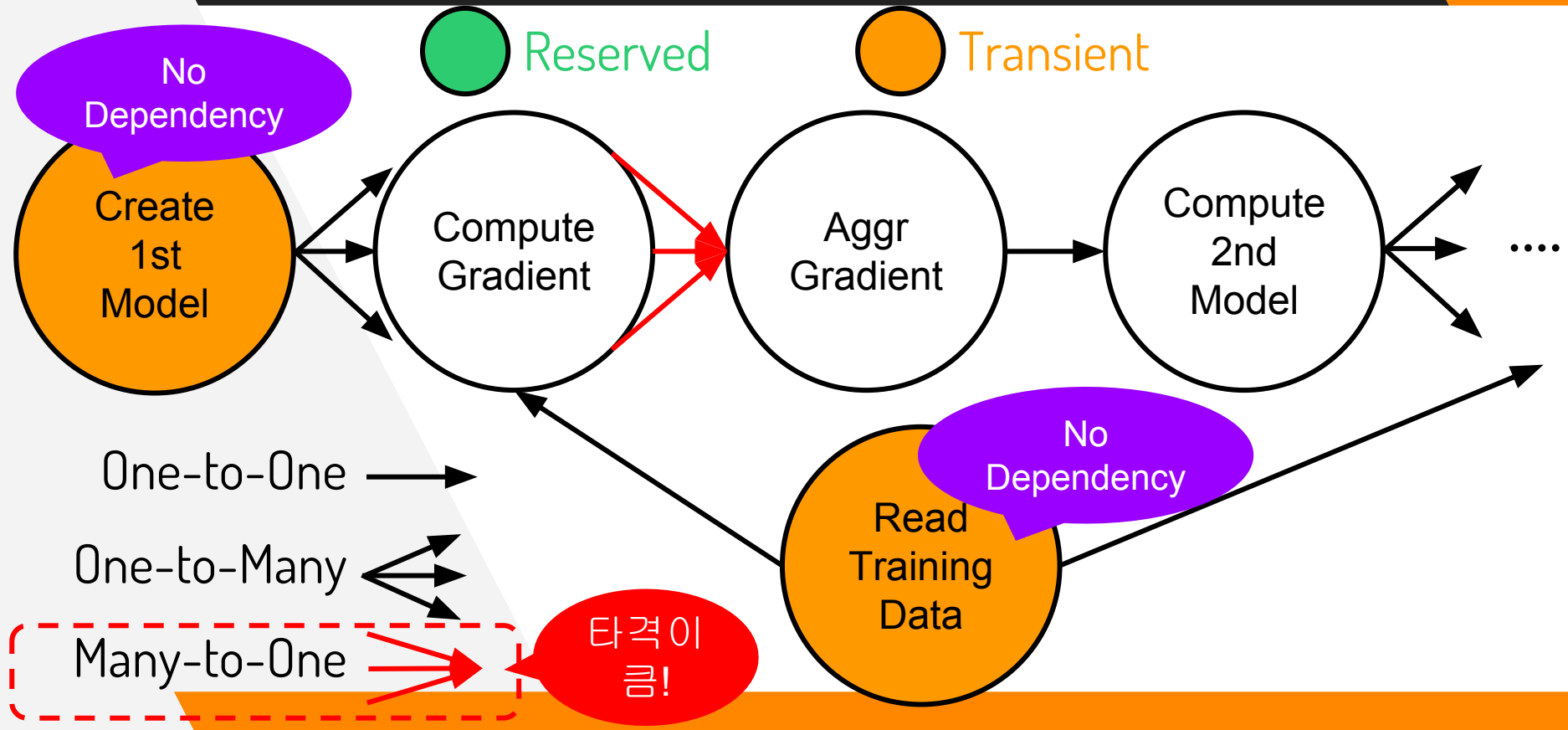
```
./bin/run_beam.sh \
      -job_id mr_default \
      -user_main org.apache.nemo.examples.beam.MinimalWordCount \
      -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount" \
      -optimization_policy org.apache.nemo.compiler.optimizer.policy.TransientResourcePolicy
```

2. 결과를 Web-UI를 사용해서 관찰

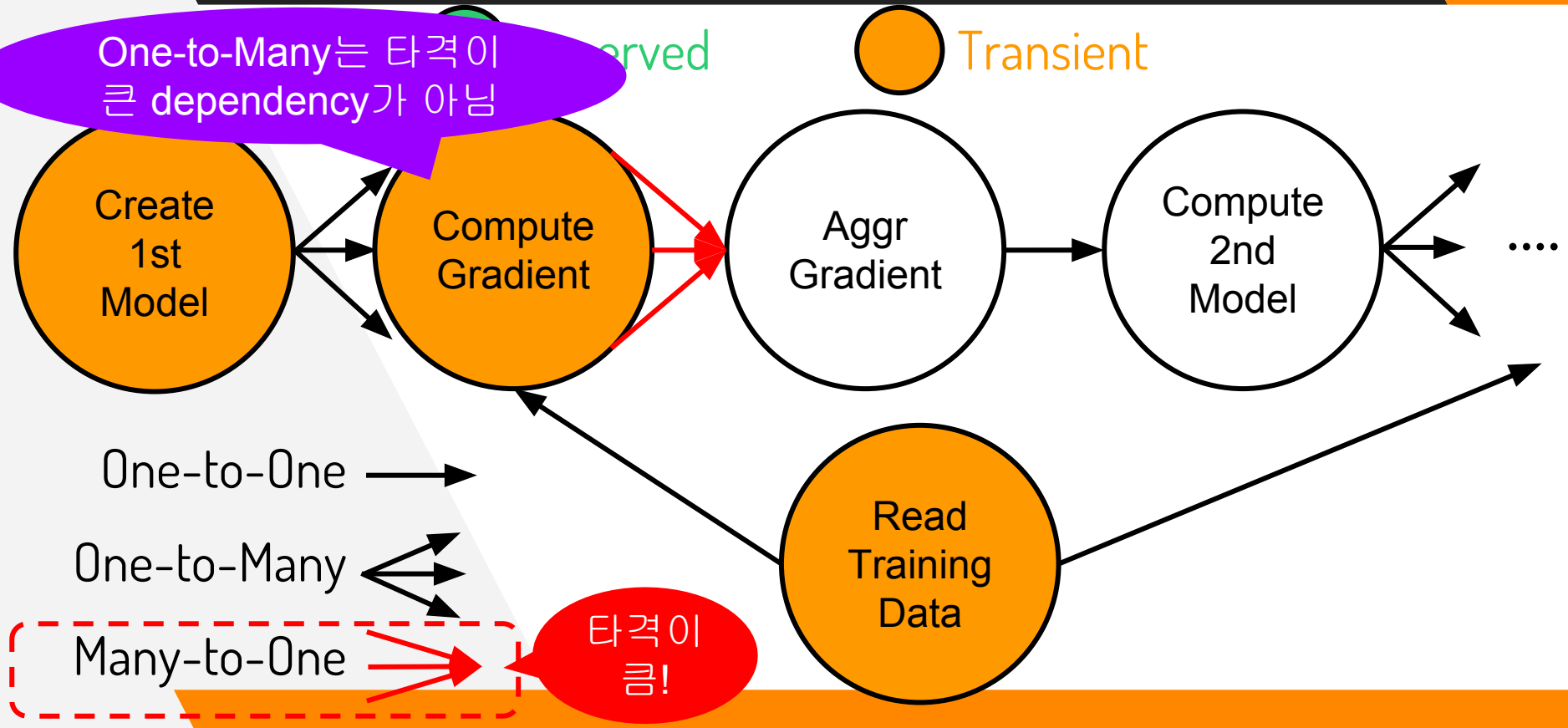
Executor Placement 예시



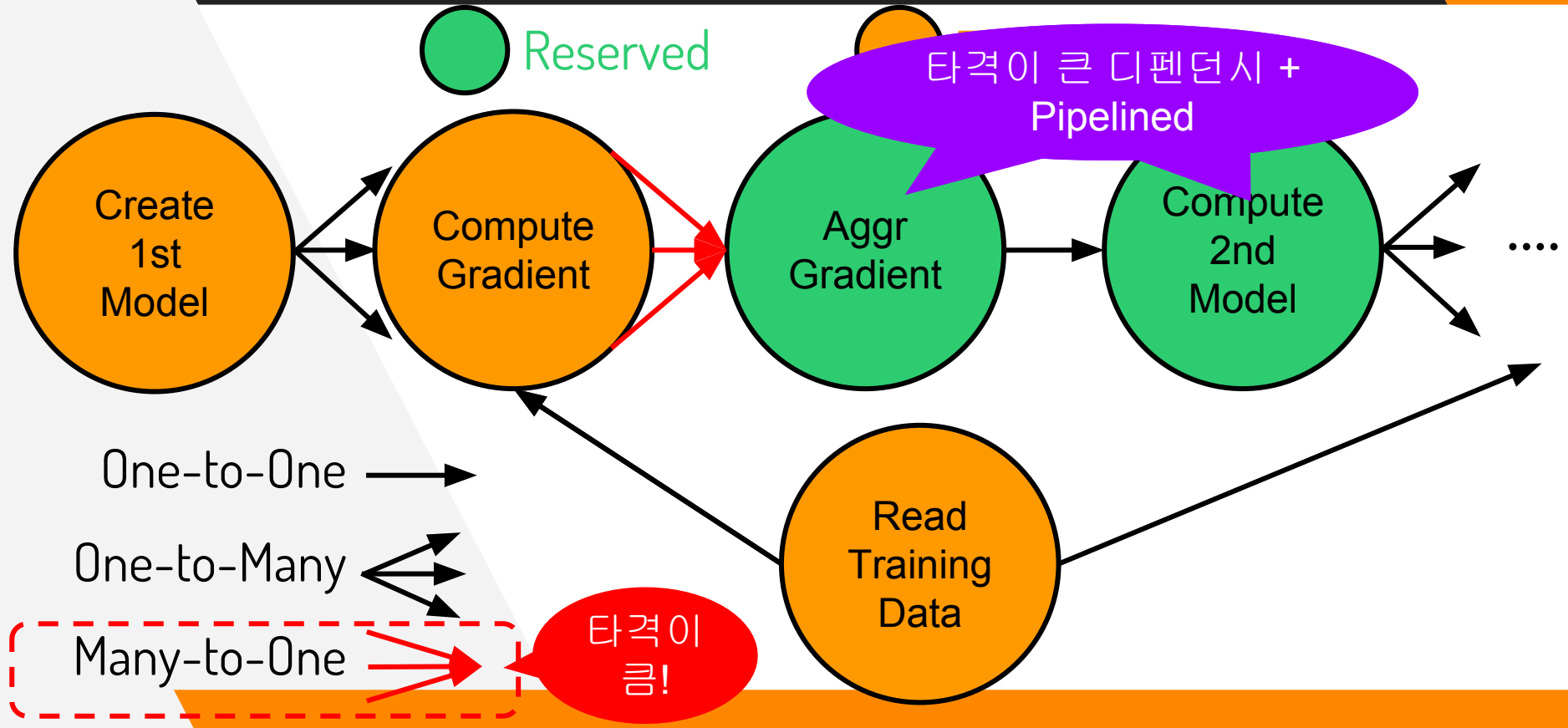
Executor Placement 예시



Executor Placement 예시

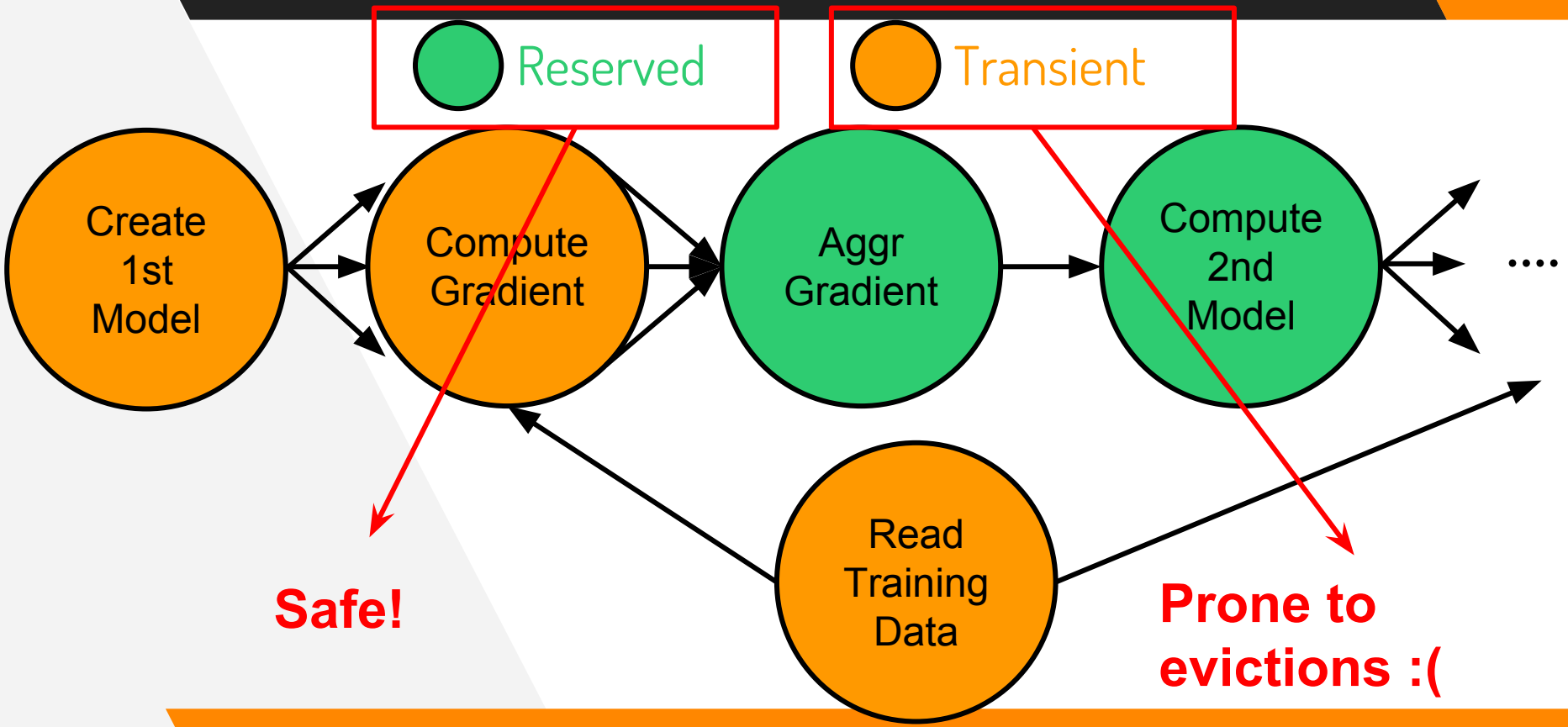


Executor Placement 예시

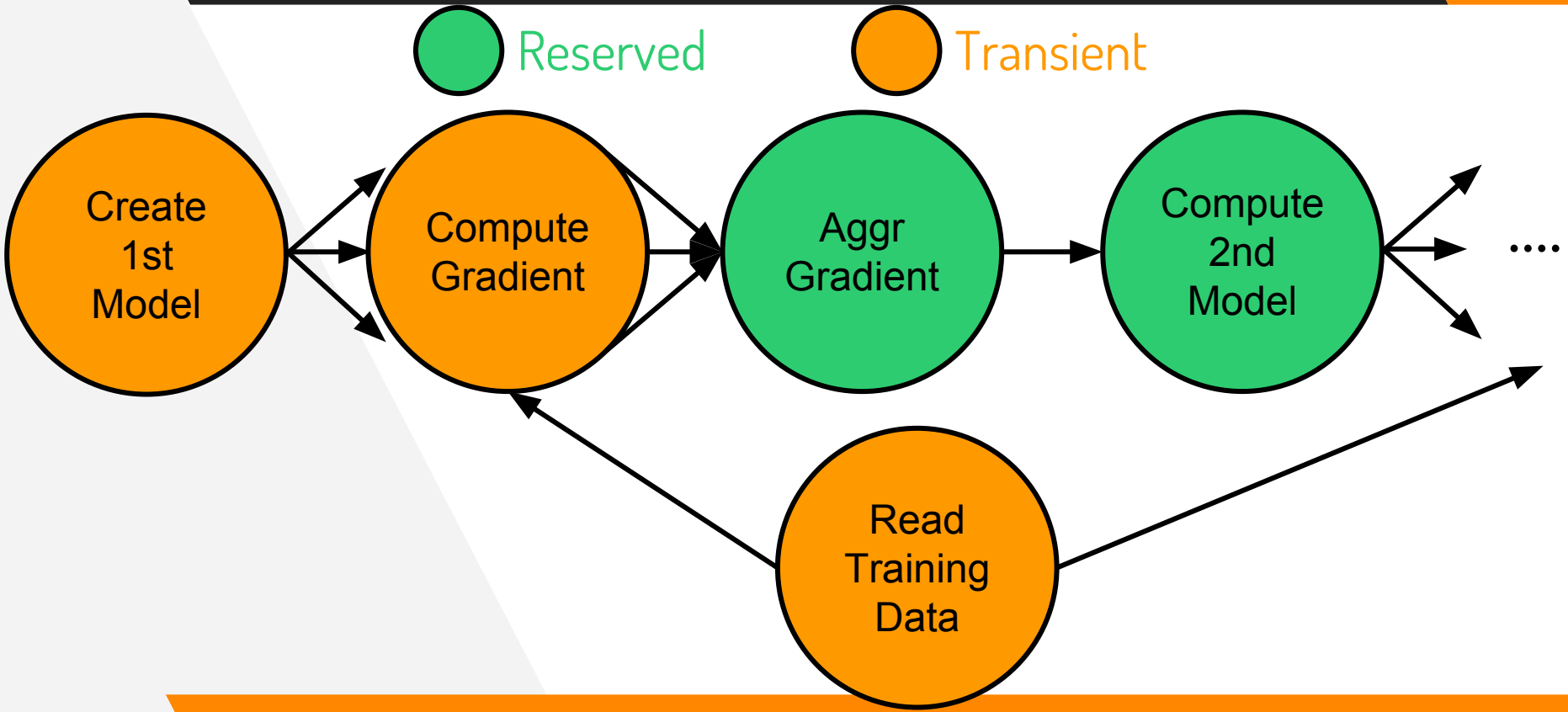


Step 2:
Data Flow Model Pass

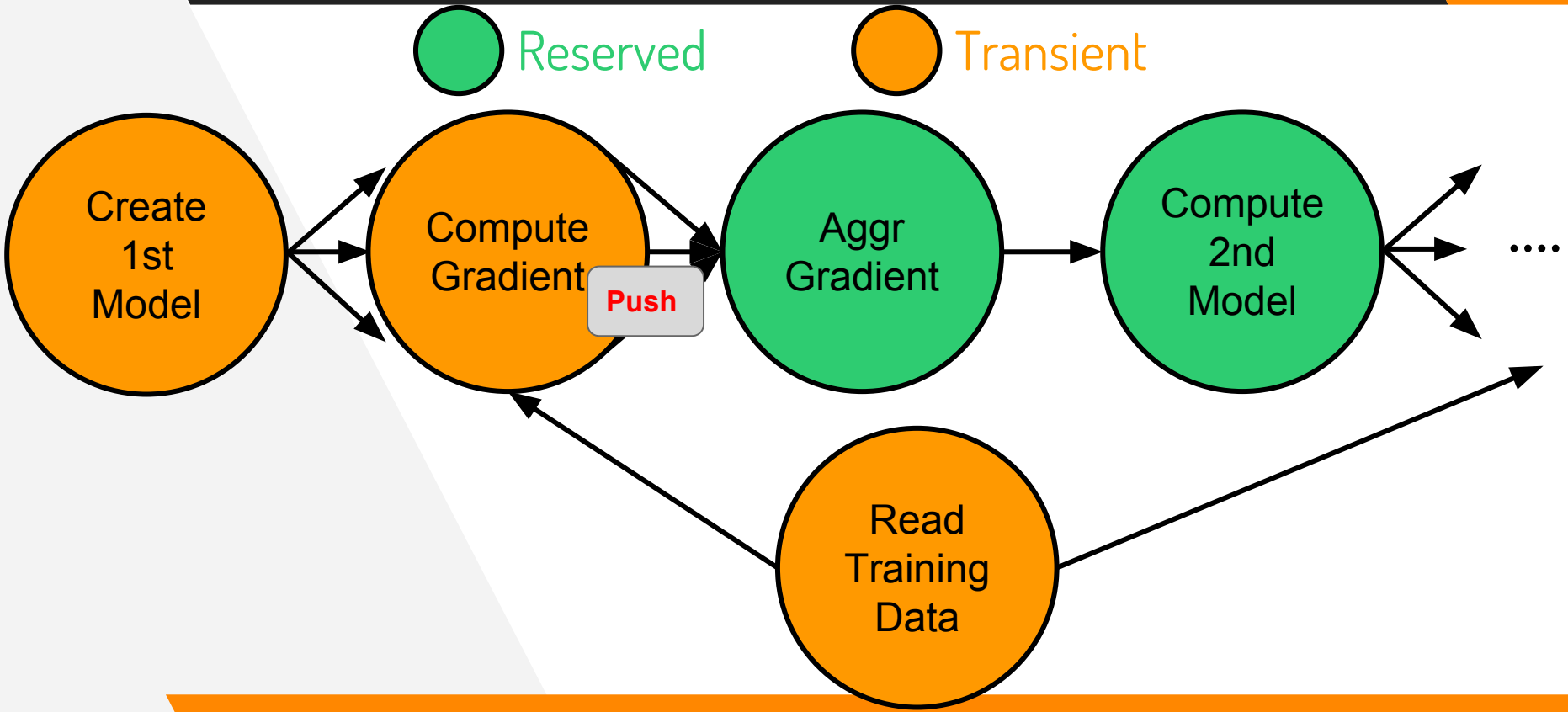
Recall..


Reserved
Transient
Create 1st Model
Compute Gradient
Aggr Gradient
Compute 2nd Model
....
Read Training Data
Safe!
Prone to evictions :(

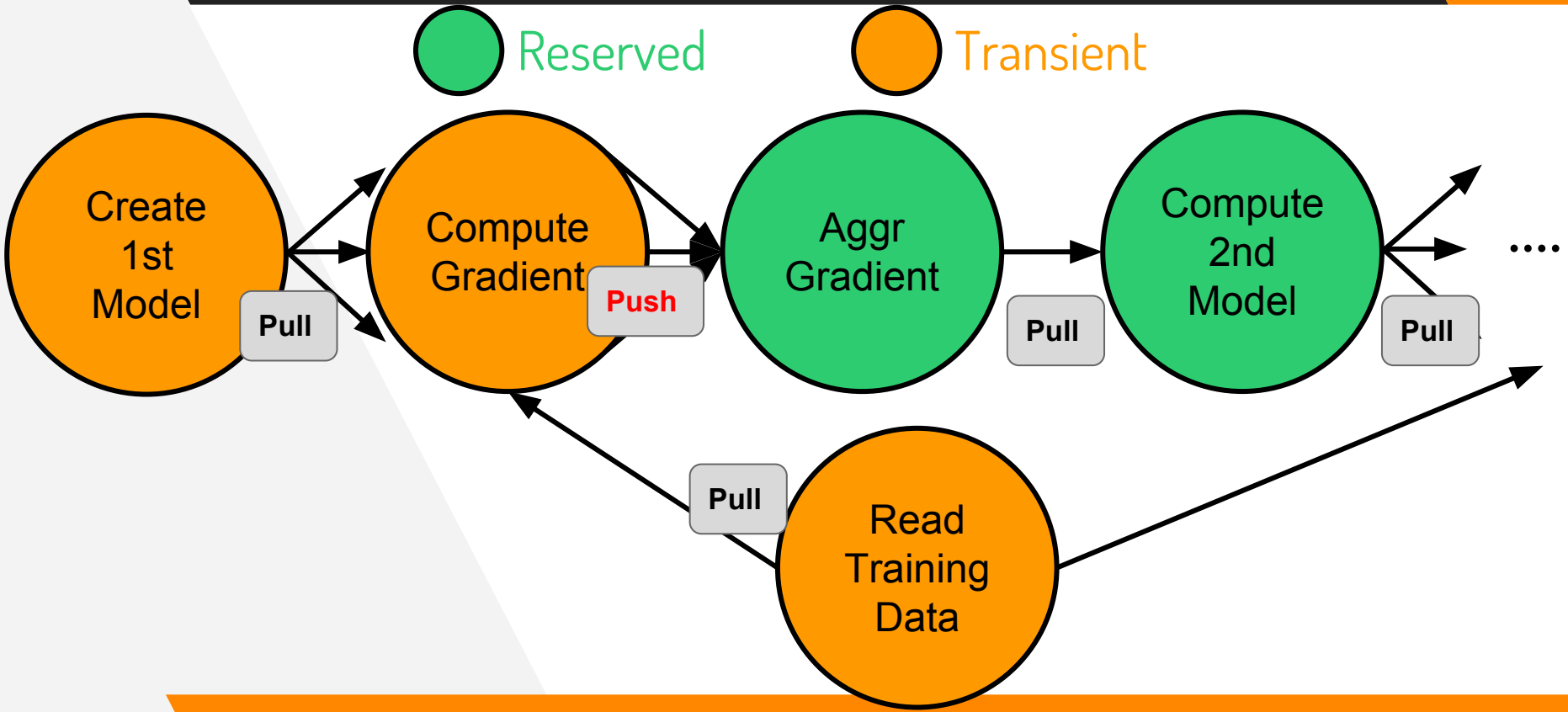
Transient executors에서 최대한 빨리 data를 옮겨야 함



Push data out as soon as it is ready!



Reserved containers에 있는 데이터는 서두를 필요 없음



TransientResourcePolicy
코드 만들어 보기

Executor Placement Pass

```
public DAG<IRVertex, IREdge> apply(final DAG<IRVertex, IREdge> dag) {
 dag.topologicalDo(vertex -> {
   final List<IREdge> inEdges = dag.getIncomingEdgesOf(vertex);
   if (inEdges.isEmpty()) {
     // #1
   } else {
     if (hasM2M(inEdges) || allO2OFromReserved(inEdges)) {
       // #2
     } else {
       // #3
     }
   }
 });
 return dag;
}
```

Executor Placement Pass

```
public DAG<IRVertex, IREdge> apply(final DAG<IRVertex, IREdge> dag) {
 dag.topologicalDo(vertex -> {
   final List<IREdge> inEdges = dag.getIncomingEdgesOf(vertex);
   if (inEdges.isEmpty()) {
     // #1
   } else {
     if (hasM2M(inEdges) || allO2OFromReserved(inEdges)) {
       // #2
     } else {
       // #3
     }
   }
 });
 return dag;
}
```

Executor Placement Pass

```
public DAG<IRVertex, IREdge> apply(final DAG<IRVertex, IREdge> dag) {
 dag.topologicalDo(vertex -> {
   final List<IREdge> inEdges = dag.getIncomingEdgesOf(vertex);
   if (inEdges.isEmpty()) {
     vertex.setProperty(ResourcePriorityProperty.of(ResourcePriorityProperty.TRANSIENT));
   } else {
     if (hasM2M(inEdges) || allO2OFromReserved(inEdges)) {
       vertex.setProperty(ResourcePriorityProperty.of(ResourcePriorityProperty.RESERVED));
     } else {
       vertex.setProperty(ResourcePriorityProperty.of(ResourcePriorityProperty.TRANSIENT));
     }
   }
 });
 return dag;
}
```

Data Flow Model Pass

```
public DAG<IRVertex, IREdge> apply(final DAG<IRVertex, IREdge> dag) {
 dag.getVertices().forEach(vertex -> {
   final List<IREdge> inEdges = dag.getIncomingEdgesOf(vertex);
   if (!inEdges.isEmpty()) {
     inEdges.forEach(edge -> {
       if (fromTransientToReserved(edge)) {
         // #1
       } else {
         // #2
       }
     });
   }
 });
 return dag;
}
```

Data Flow Model Pass

```
public DAG<IRVertex, IREdge> apply(final DAG<IRVertex, IREdge> dag) {
 dag.getVertices().forEach(vertex -> {
   final List<IREdge> inEdges = dag.getIncomingEdgesOf(vertex);
   if (!inEdges.isEmpty()) {
     inEdges.forEach(edge -> {
       if (fromTransientToReserved(edge)) {
         edge.setProperty(DataFlowProperty.of(DataFlowProperty.Value.Push));
       } else {
         edge.setProperty(DataFlowProperty.of(DataFlowProperty.Value.Pull));
       }
     });
   }
 });
 return dag;
}
```

Sailfish 설정을 적용한 예시

1. 다음 코드 실행

LargeShufflePolicy 적용

```
./bin/run_beam.sh \
      -job_id mr_default \
      -user_main org.apache.nemo.examples.beam.MinimalWordCount \
      -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount" \
      -optimization_policy org.apache.nemo.compiler.optimizer.policy.LargeShufflePolicy
```

2. 결과를 Web-UI를 사용해서 관찰

LargeShufflePolicy
코드 살펴보기

Pado 와 Sailfish 설정을 동시에 적용한 예시

1. 다음 코드 실행

두 optimization이 동시에 들어간 Policy 적용

```
./bin/run_beam.sh \
      -job_id mr_default \
      -user_main org.apache.nemo.examples.beam.MinimalWordCount \
      -user_args "`pwd`/LICENSE `pwd`/test_output_wordcount" \
      -optimization_policy
org.apache.nemo.compiler.optimizer.policy.TransientResourceAndLargeShufflePolicy
```

2. 결과를 Web-UI를 사용해서 관찰

TransientResourceAndLargeShufflePolicy 코드 살펴보기

시연 및 마무리

Large scale 컴퓨팅 클러스터 아래에서 실행 DEMO

(동영상)

Apache Nemo는 오픈소스 입니다!
언제든 Contribute 해 주세요! 😊

홈페이지: http://nemo.apache.org/
깃헙: https://github.com/apache/incubator-nemo

Thank you!