分散フレームワークChristieの設計

- 当研究室ではスケーラブルな分散プログラムを信頼性高く記述できる環境を実現するために並列分散フレームワークAliceを開発している

- Aliceの通信の信頼性を高めるためにNAT越えの機能設計を行ったが、その実現にはAliceの再設計が必要であった

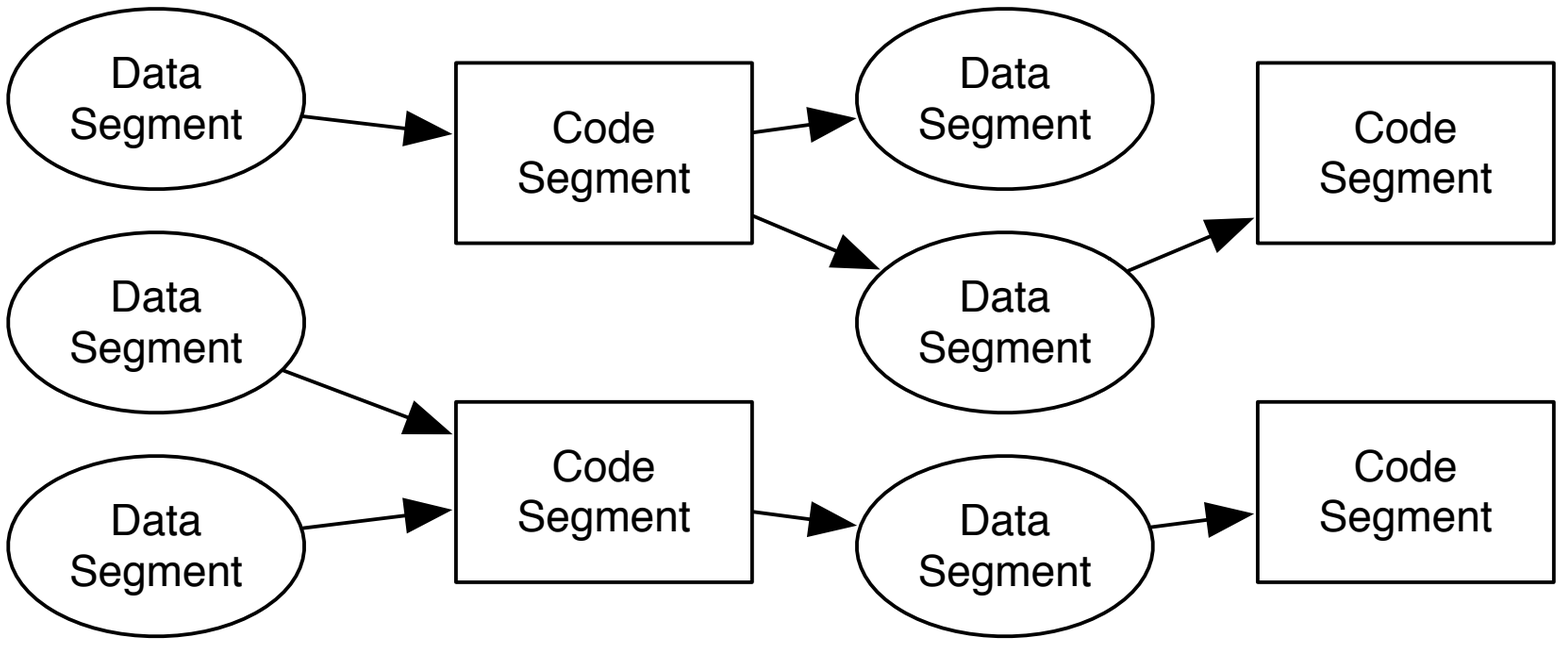
- 本研究ではAliceの問題点を整理し、得られた知見をもとに分散フレームワークChristieの設計を行う

分散フレームワークAlice

- AliceではデータをData Segment(DS)、タスクをCode Segment(CS)という単位に分割して依存関係を記述することでプログラミングを行う。

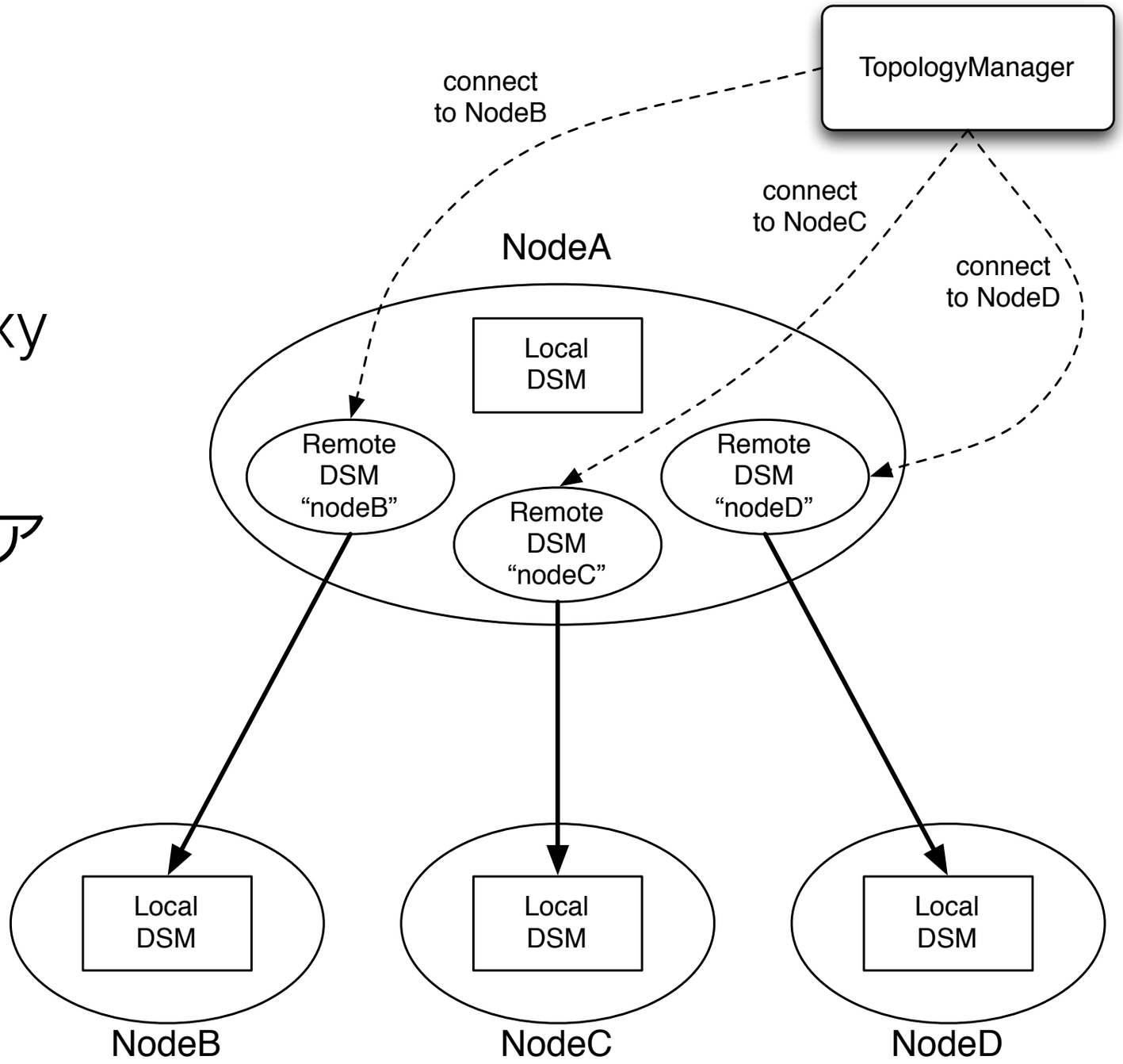
- CSはInput DS(入力されるDS)とOutput DS(出力されるDS)を持ち、keyで指定したInputDSが全て揃うと実行される

- データの依存関係にないCSは並列実行される



分散フレームワークAlice

- DSはDSManagerが管理
- 他ノードのDSMへはproxyであるRemote DSMを立ててDSMkeyを指定してアクセス
- TopologyManagerがトポロジー構成をサポート



Aliceの問題点(1)
DSMインスタンスを複数作れない

- Local DSMを複数立ち上げられれば現状のTopologyManagerのコードを変えずにNAT越えなどが実装可能

- Local DSMを複数立ち上げられないため、Topology Managerの拡張や分散テストが困難

Aliceの問題点(2)

APIシンタックスの分離

- inputDSはcreateメソッドでReceiverを生成して、
  その後setKeyメソッドでkeyを指定。

- 記述が分離しているため、CSを見てもどんなkeyのデータを待ち合わせているのか分からないことがある

- create/setKeyの書く順序によっては正しい挙動で動かない

```
class TestCG extends CodeSegment{
    private Receiver input1 = ids.create(CommandType.TAKE);

    void run() {
        System.out.println(input1.asClass(Integer.class));
        CodeSegment cg = new TestCG();
        cg.input1.setKey("hoge");
    }
}
```

Aliceの問題点(3)
型が推測できない

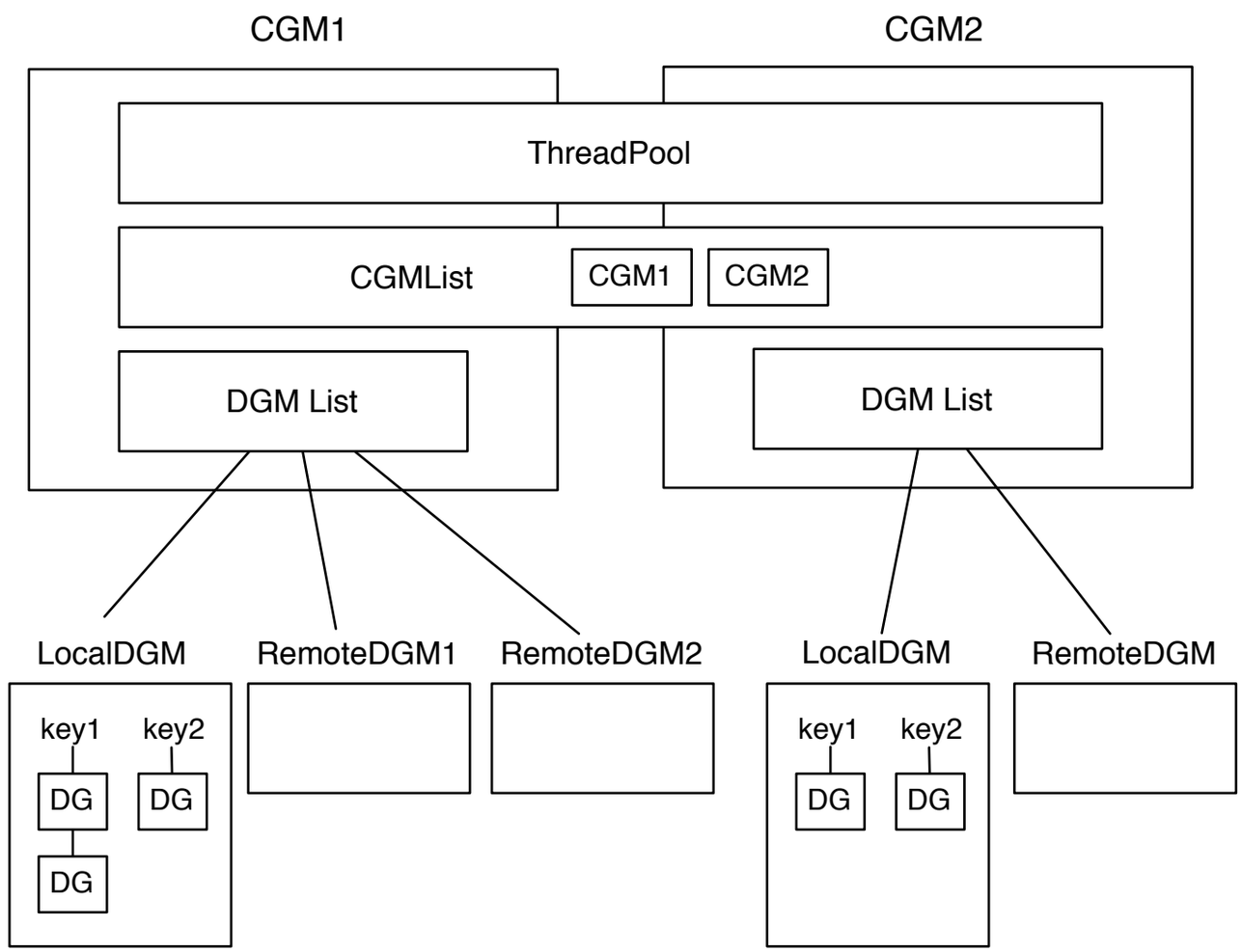
- Receiver型で生成するため、CSを見ただけでは
  どんな型を待ち合わせているかわからない
- InputDSからデータを取り出すためには型の指定が必要

```
class TestCG extends CodeSegment{
    private Receiver input1 = ids.create(CommandType.TAKE);

    void run() {
        System.out.println(input1.asClass(Integer.class));
        CodeSegment cg = new TestCG();
        cg.input1.setKey("hoge");
    }
}
```

Christieの基本設計

- CodeGear/DataGearでプログラミング
- CodeGearManagerがDataGearManagerを管理
- LocalDGMを複数立ち上げられる



アノテーションを用いた
インプットの記述

- アノテーションを用いてTake/Peek、DSMkeyを指定する
- 変数名がそのままkeyとして扱われる
- InputDSの生成、key指定、Take/Peekの指定の分離を防ぐ

```
@Take
String name;

@TakeFrom("remote")
int count;
```

アノテーションを用いた
インプットの記述

- 型をそのまま書くためわかりやすい
- 型を内部で保存するため型を指定しての取り出しが必要ない

```
@Take
String name;

@TakeFrom("remote")
int count;

public void run(CodeGearManager cgm){
    System.out.println(count + " : " + name);
    cgm.setup(new TestCodeGear());
}
```

まとめ

- Aliceからの知見を元にChristieを設計した
- CodeGearManagerというDGMの管理機構を作ったことでLocalDGM複数立ち上げが可能になり、NAT越えなどの機能拡張やテストをしやすくなった
- アノテーションを用いたことでDG生成とkey指定の分離問題を解決し、処理の見通しを良くした
- 型の整合性を保証することで信頼性が向上した