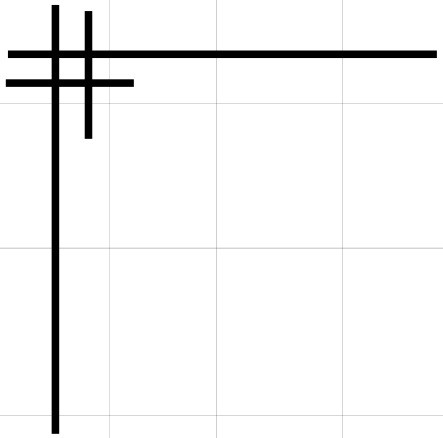
現役高校生が実装する Skip Graph with Erlang

千々和 大輝（@_daiki)

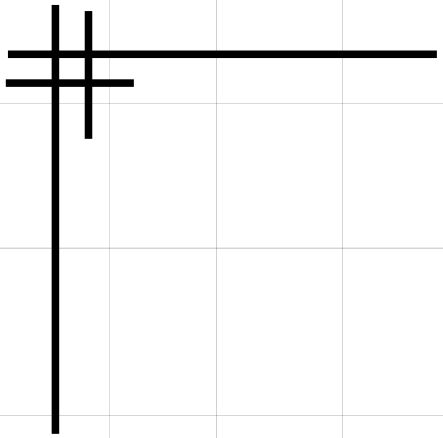
自己紹介

- 千々和 大輝
- P2P を応用した分散処理
  - P2P 匿名掲示板
  - key-value store
- Twitter: _daiki



Skip Graph

Erlang



Skip Graph

Erlang

P2P

- 構造化オーバーレイ
  - 規則に従って構造化されたネットワーク
  - 分散ハッシュテーブルや Skip Graph など
- 非構造化オーバーレイ
  - 初期の P2P
  - Gnutella や Winny など

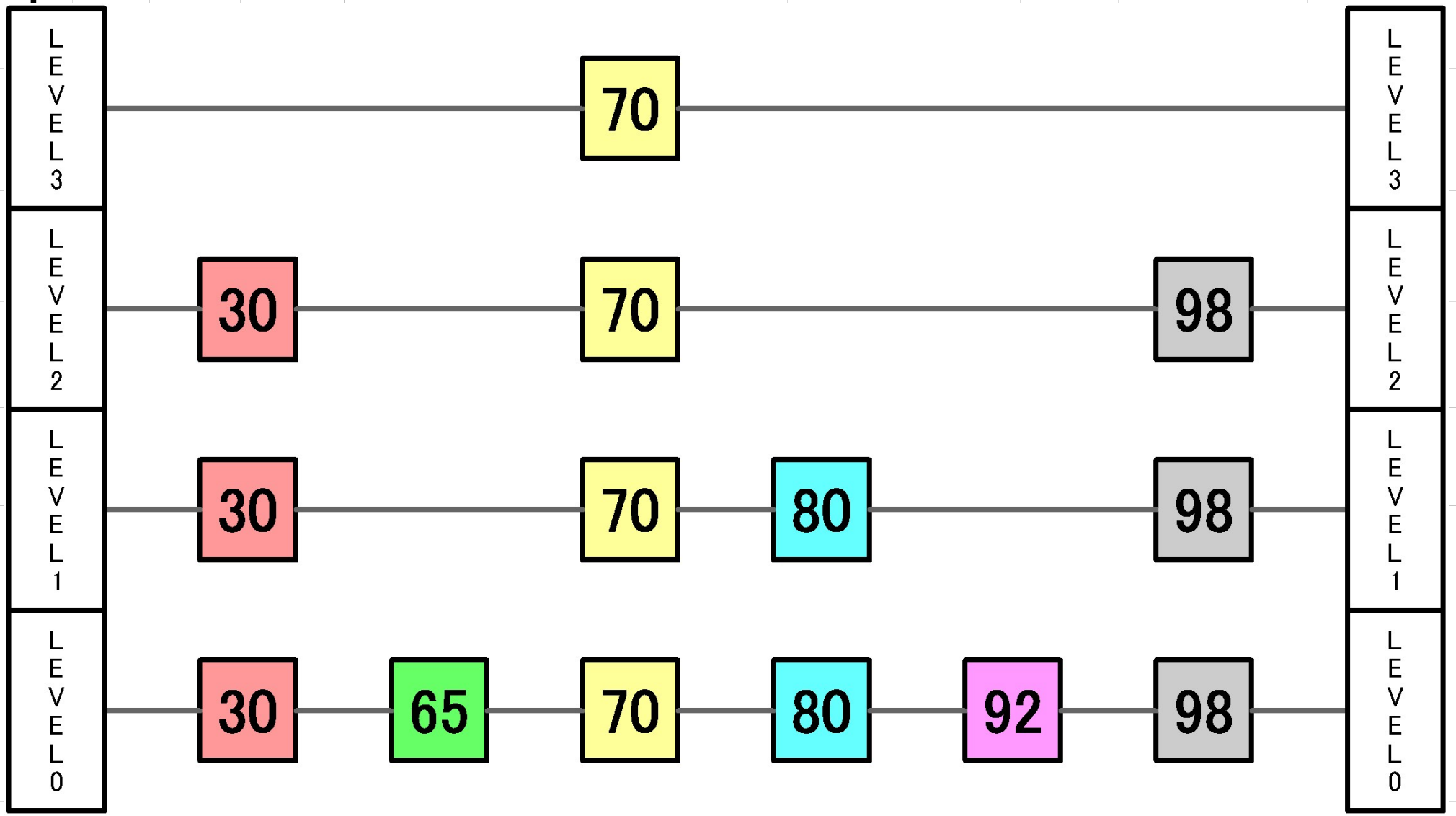
分散ハッシュテーブル (DHT)

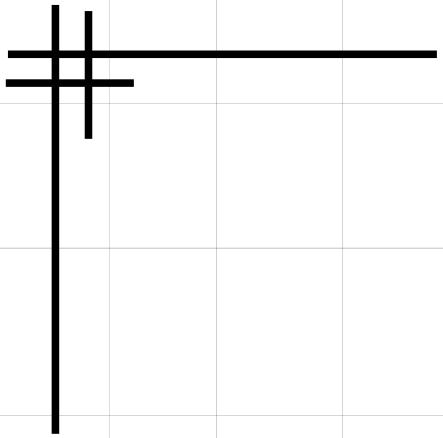
- 分散したハッシュテーブル
  - 基本は一般的なハッシュテーブルと同じ
  - ハッシュ値を利用して探索 / 挿入する
  - 範囲検索が不可能

Skip Graph とは？

- スキップリストアルゴリズムが元
- DHT とは異なり,キーはそのまま利用される
  → 順序性が保たれる
  → キーによる範囲検索が可能に!
- 挿入 / 探索速度は DHT に劣らず

Skip List


LEVEL3
LEVEL2
LEVEL1
LEVEL0
70
30
70
98
30
70
80
98
30
65
70
80
92
98
LEVEL3
LEVEL2
LEVEL1
LEVEL0



Skip Graph

Erlang

Erlang とは？

- 並行指向の関数型言語
- 軽量プロセス
- 透過的な分散処理
- 強力な耐障害性を持つ

OTP(Open Telecom Platform)

- 似たようなコードの簡略化
- 汎用的な動作 (behaviour) のライブラリ
- 様々な behaviour
  - supervisor
  - gen_server
  - gen_event
  - etc.....

コード例

```
1 -module(qsort).
2 -export([qsort/1]).
3
4 qsort([]) ->
5     [];
6 qsort([H | T]) ->
7     qsort1([ X || X <- T, X < H ]) ++ [H] ++ qsort1([ X || X <- T, X >= H ]).
```

説明おわり

Erlang/OTP による恩恵

- OTP (gen_server)
  - 障害発生時は自動的に再起動
  - 透過的な分散処理
  - コードの一貫性

Concurrent Join の実装

- 複数の Join 処理を同期的に実行
  - 複雑
  - バグが入りやすい
  - 不整合の発生をなるべく防ぐ
  - デッドロックの恐れアリ

今後の課題

- Concurrent Join の改良
- あるノードがいきなり死んだ場合の対処
- より多くの key-value ペアの保持を可能に
- C++ に移植

ありがとうございました