はじめての Droonga

Droongaの簡単な紹介と Groongaからの移行手順

結城洋志

株式会社クリアコード

要旨

「自作のアプリケーションを
GroongaからDroongaへ
今すぐ移行できるのか？」
にお答えします

気になる点、疑問点

- 気になる事があったら:
  - メモして後から質問
  - その場で質問してもOK
- どこが気になったかを
  教えてください!

アジェンダ

- Droongaとは？
  - Droongaの何が嬉しい？
  - Droongaの何が嬉しくない？
- デモ

Groongaの困った所

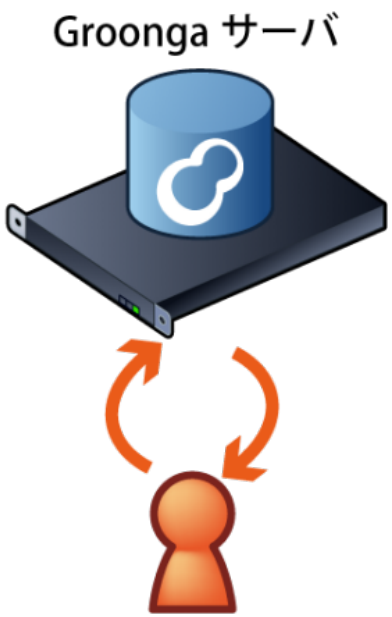
- 分散が流行ってる
- Groongaは分散処理に対応していない

Droongaとは

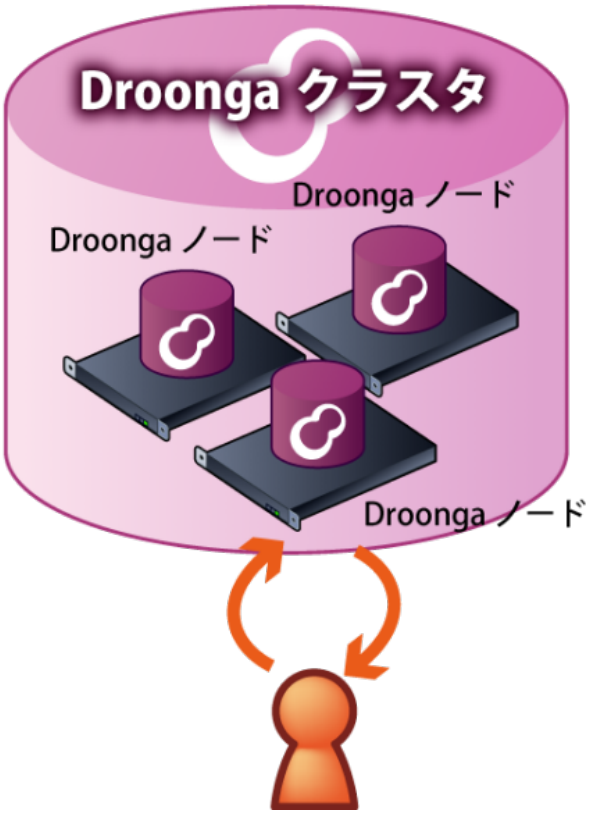
Distributed
Groonga
＝分散Groonga

サーバ構成の違い

groonga



droonga



Groonga互換


groonga
droonga
Droonga クラスタ
Groonga サーバ
Droonga ノード
Droonga ノード
Droonga ノード
HTTP
HTTP
互換性あり

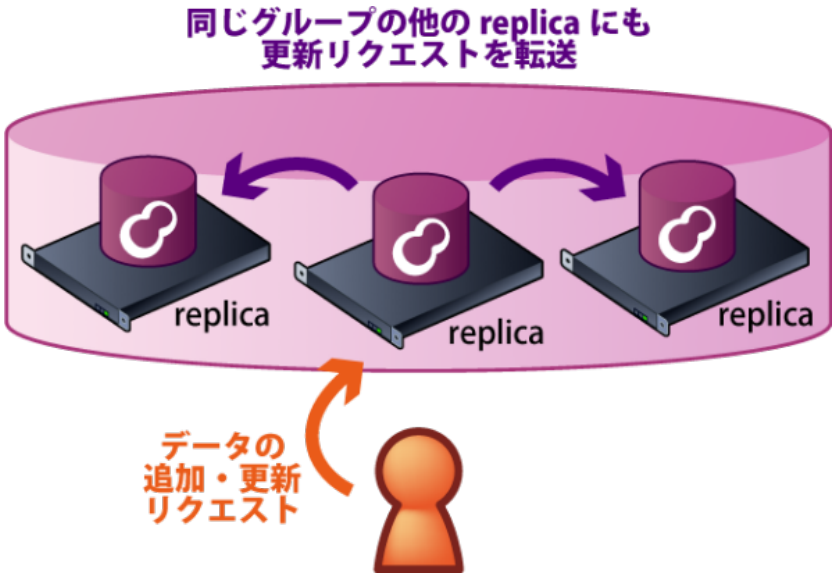
Groonga互換

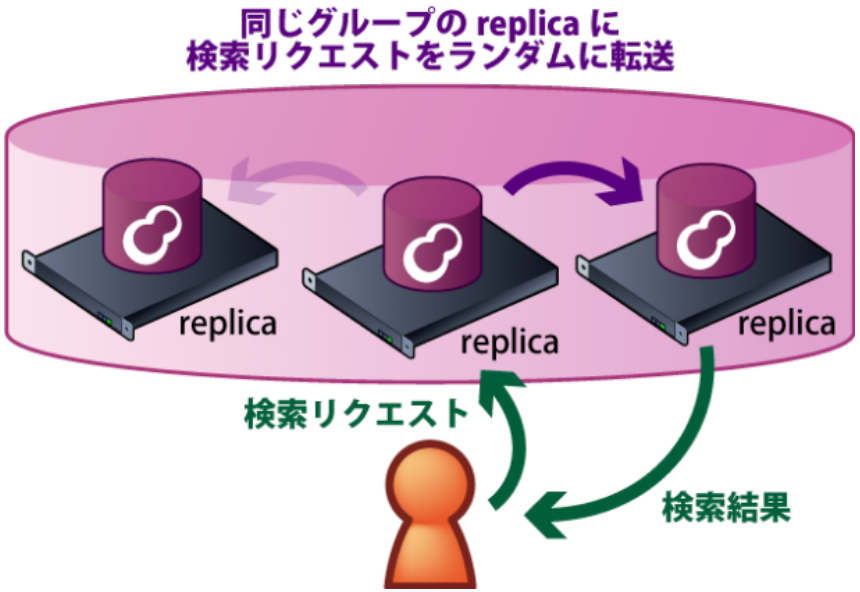
- 今までと同じ感覚で使える
- Groongaベースの
  既存のアプリケーションを
  最小の工数で分散対応できる

データベースを分散

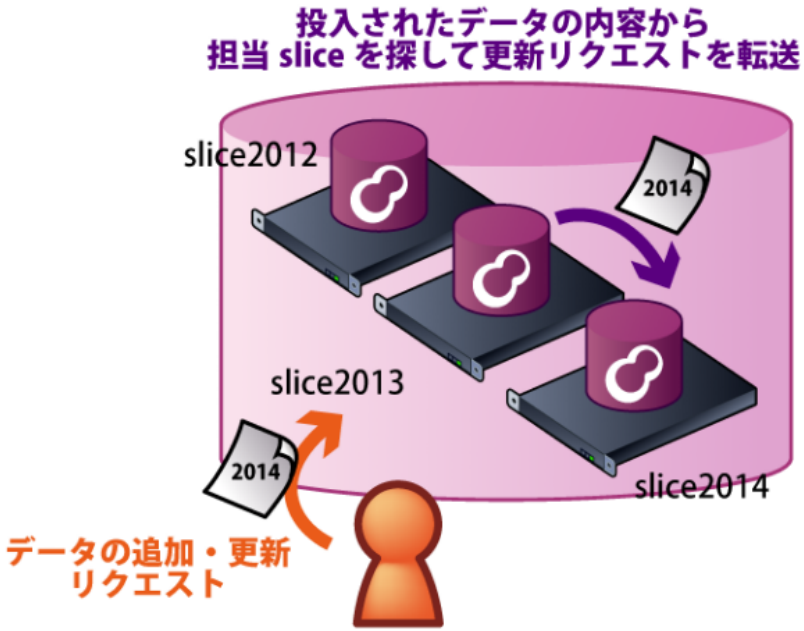
- レプリケーション
  - 現在の開発はここに注力
- パーティショニング
  - 現在は部分的に対応（これから改善）

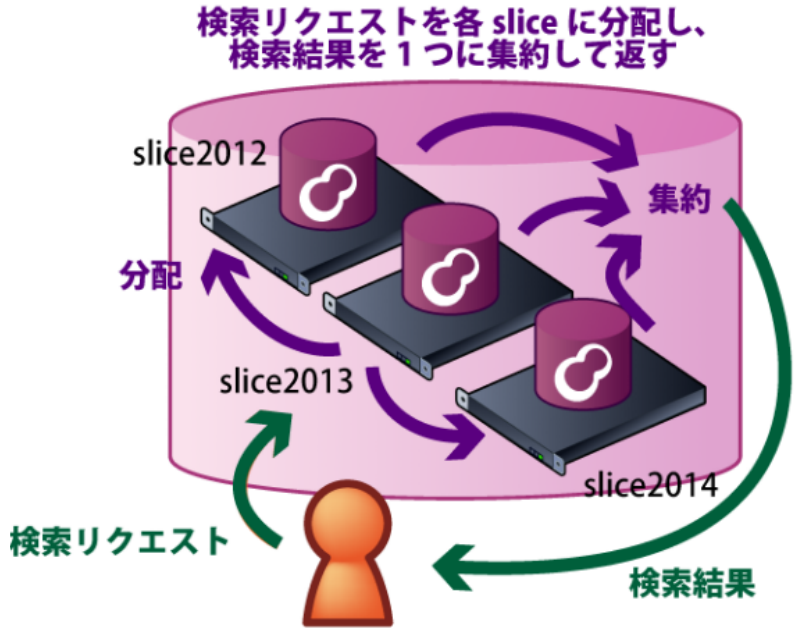
レプリケーション？


同じグループの他の replica にも
更新リクエストを転送
replica
replica
replica
データの
追加・更新
リクエスト


同じグループの replica に
検索リクエストをランダムに転送
replica
replica
replica
検索リクエスト
検索結果

パーティショニング?


投入されたデータの内容から
担当 slice を探して更新リクエストを転送
slice2012
2014
slice2013
2014
slice2014
データの追加・更新
リクエスト

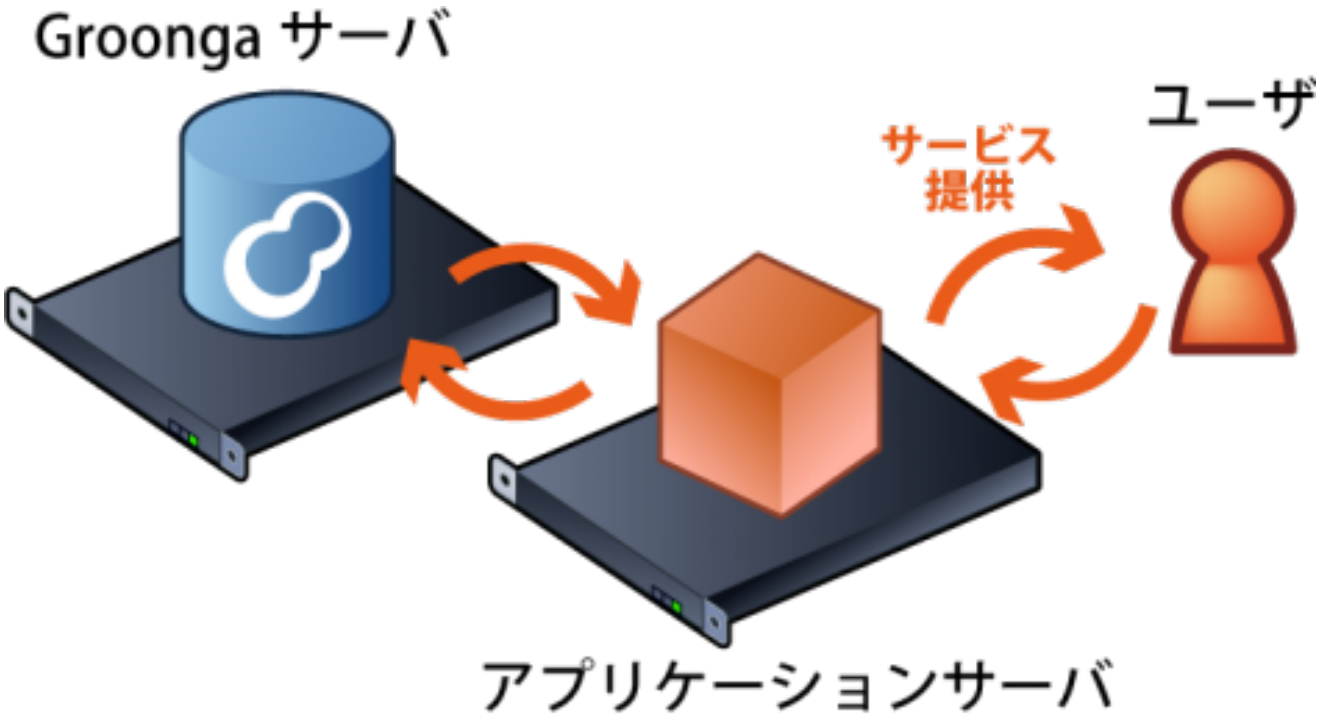

検索リクエストを各 slice に分配し、
検索結果を 1 つに集約して返す
slice2012
集約
分配
slice2013
slice2014
検索リクエスト
検索結果

Groonga→Droonga

今現在得られるメリット

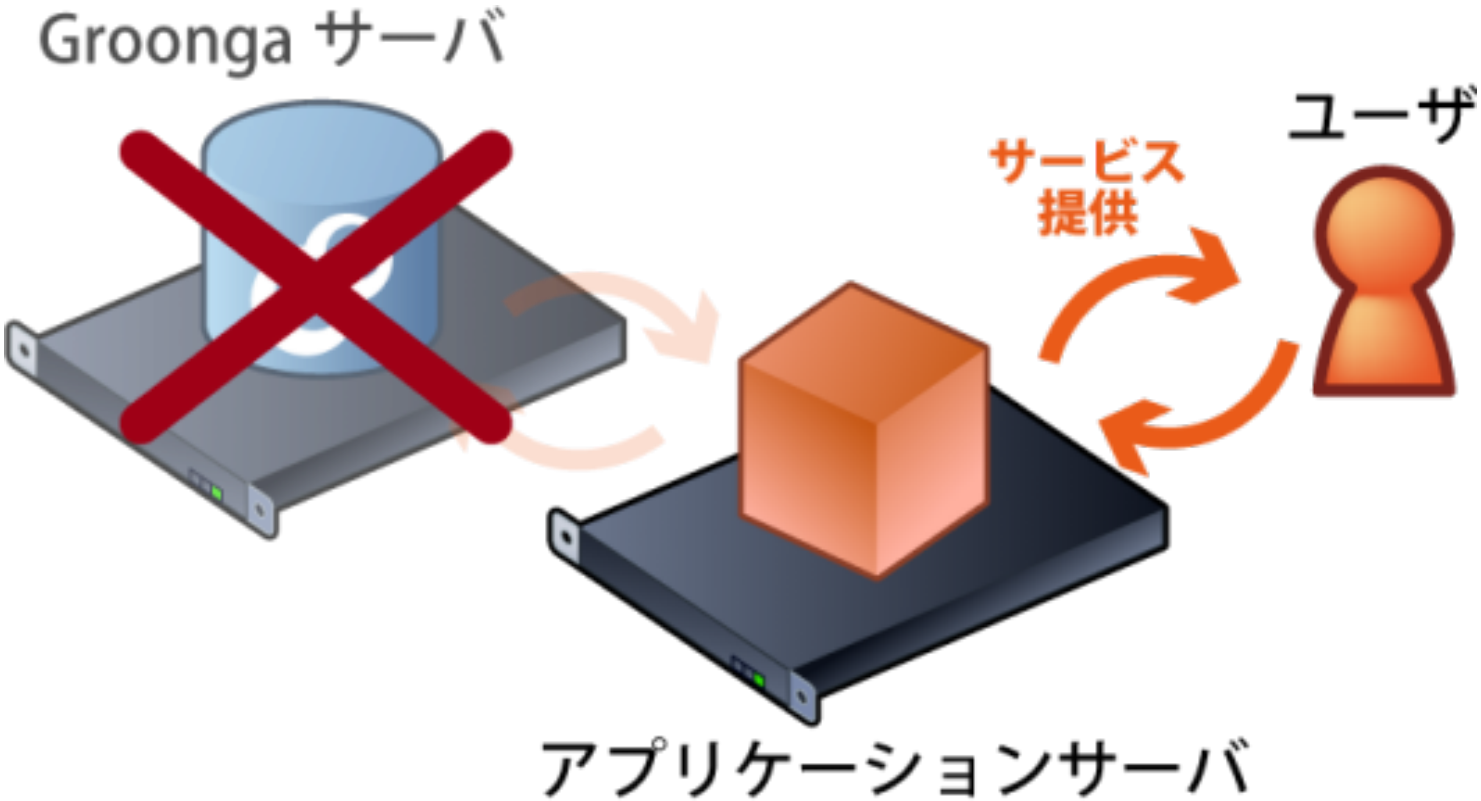
- **レプリケーションできる**
  ようになる
- ノードを**簡単に追加・削除**できる

レプリケーション無しだと(1)



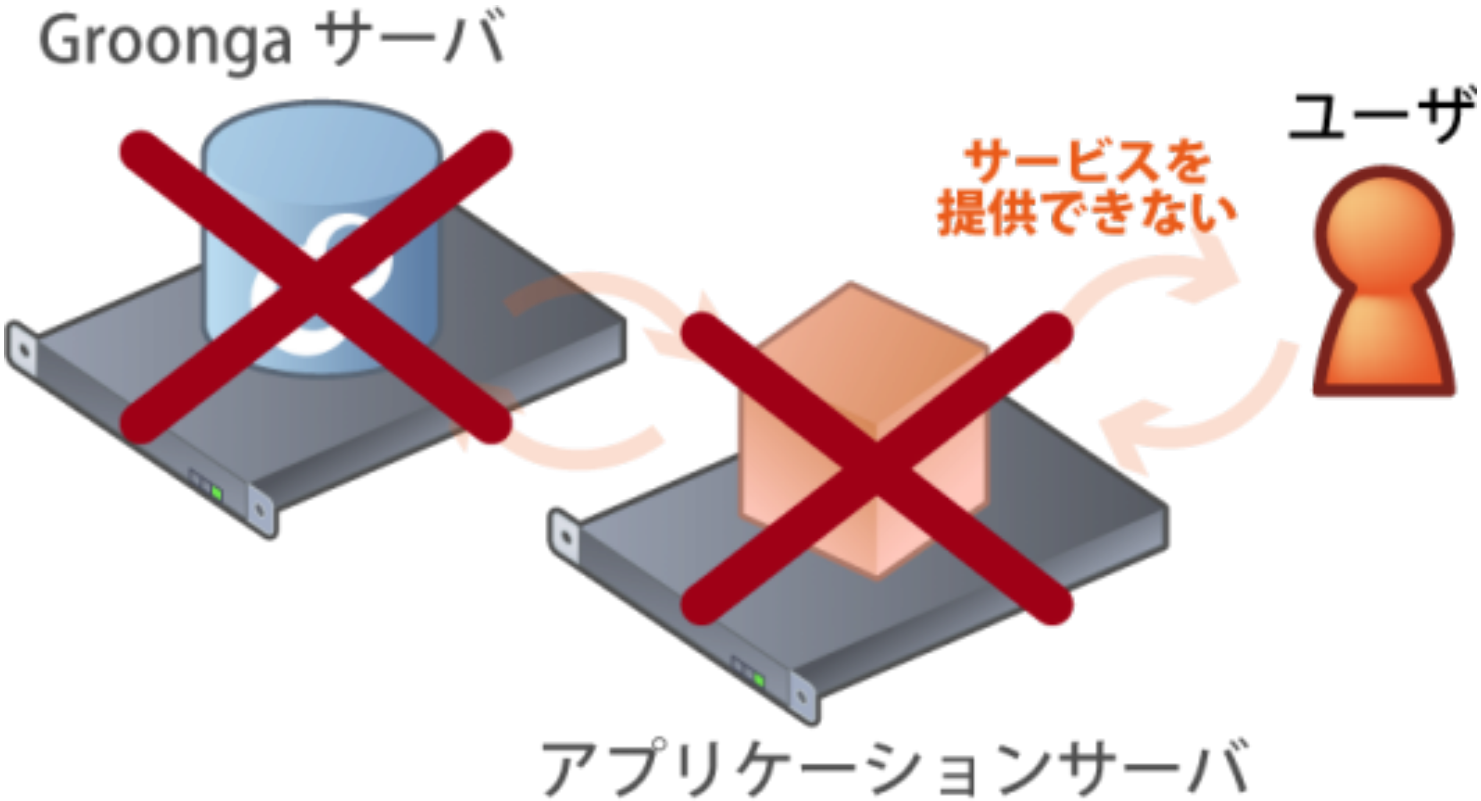
単一のGroongaサーバに
サービスが依存

レプリケーション無しだと(1)



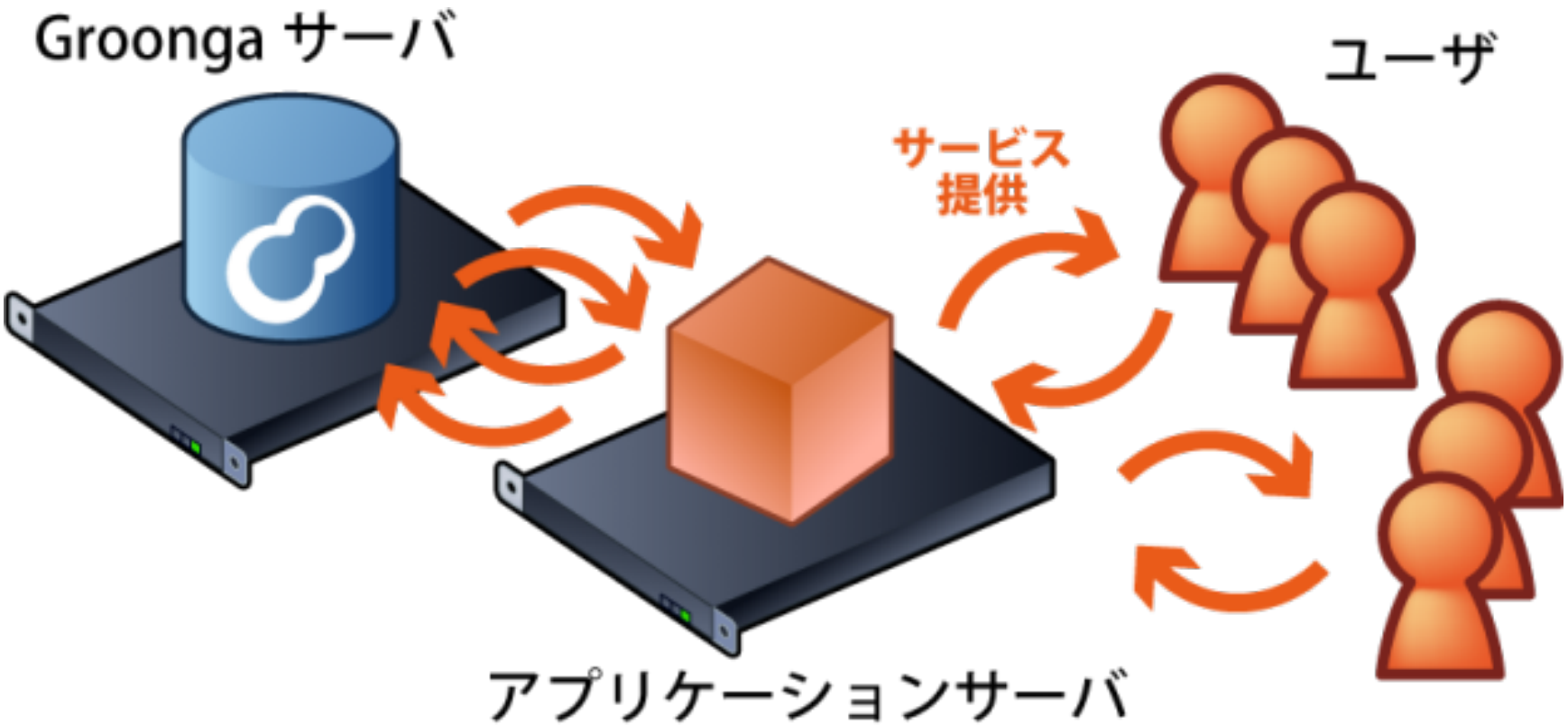
Groongaが死ぬと……

レプリケーション無しだと(1)



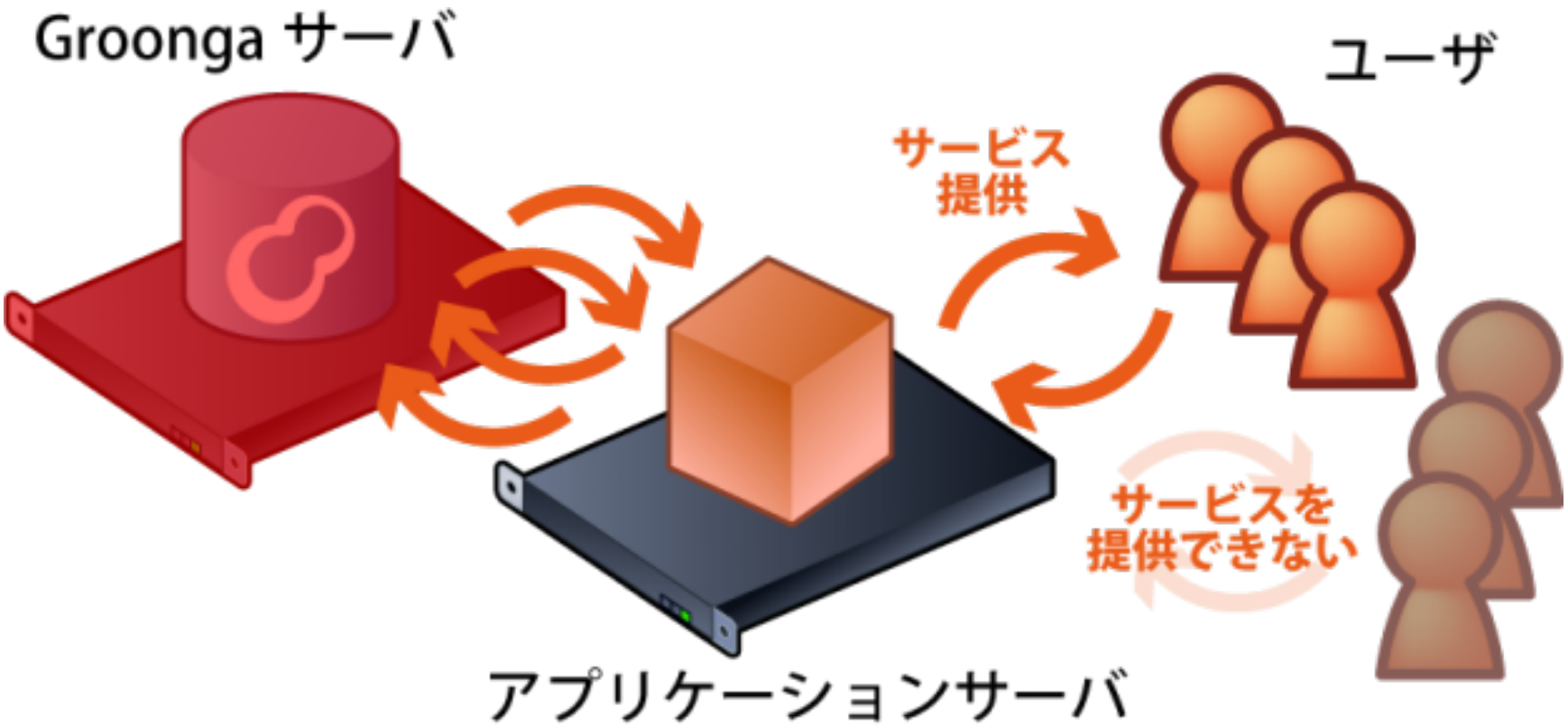
サービスも道連れになる

レプリケーション無しだと(2)



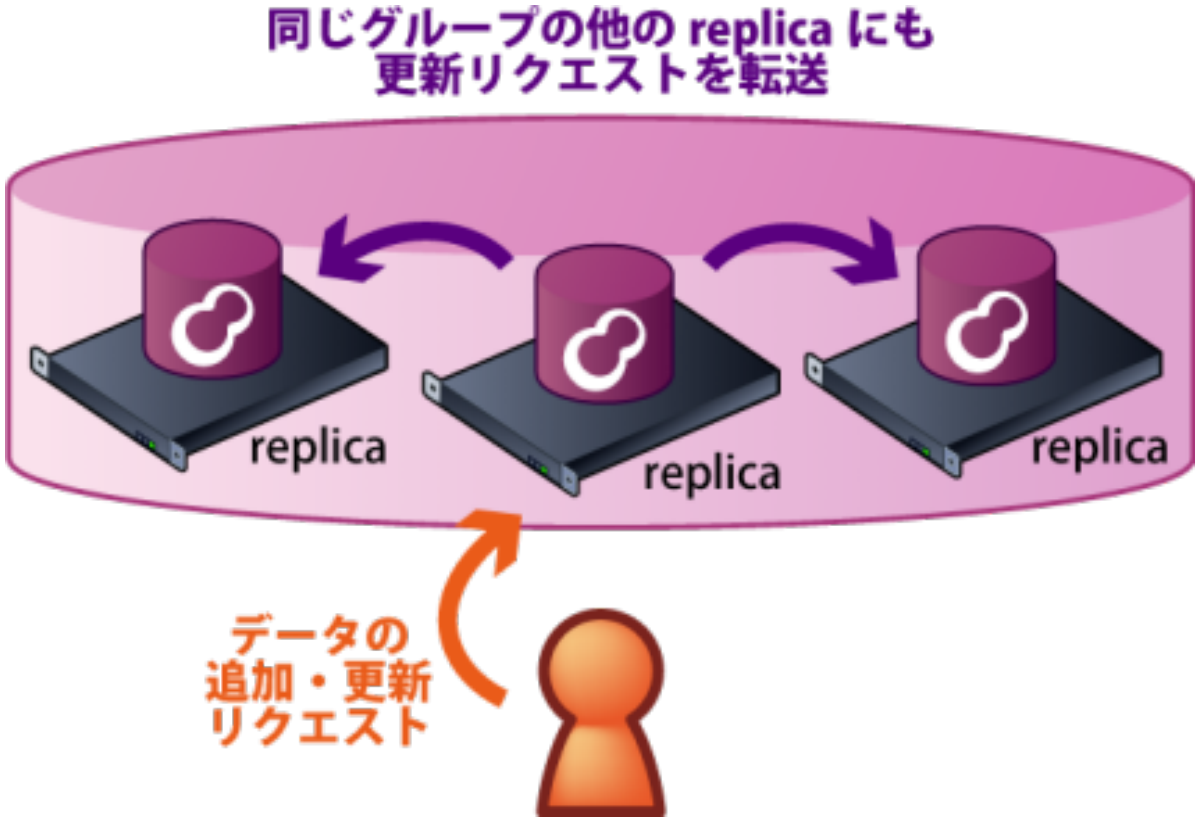
負荷が増大すると……

レプリケーション無しだと(2)



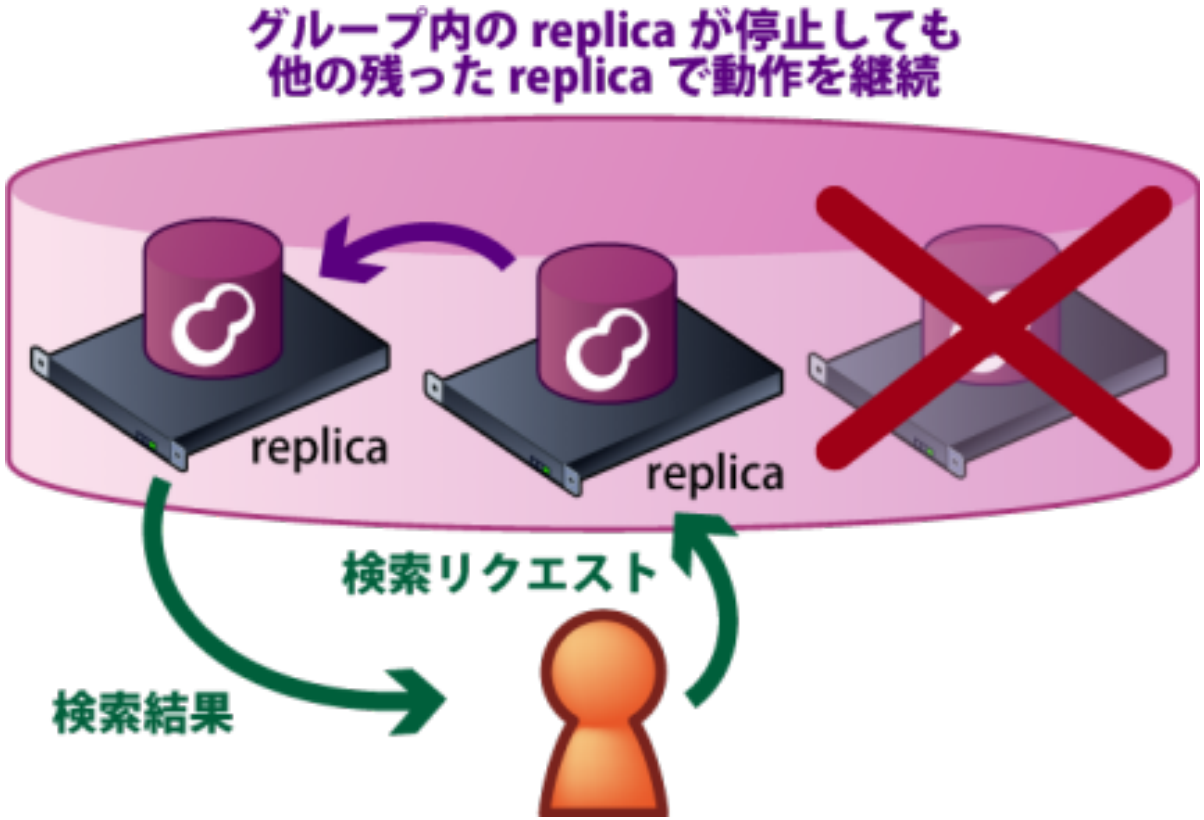
サービスレベルが落ちる

レプリケーション有りだと(1)



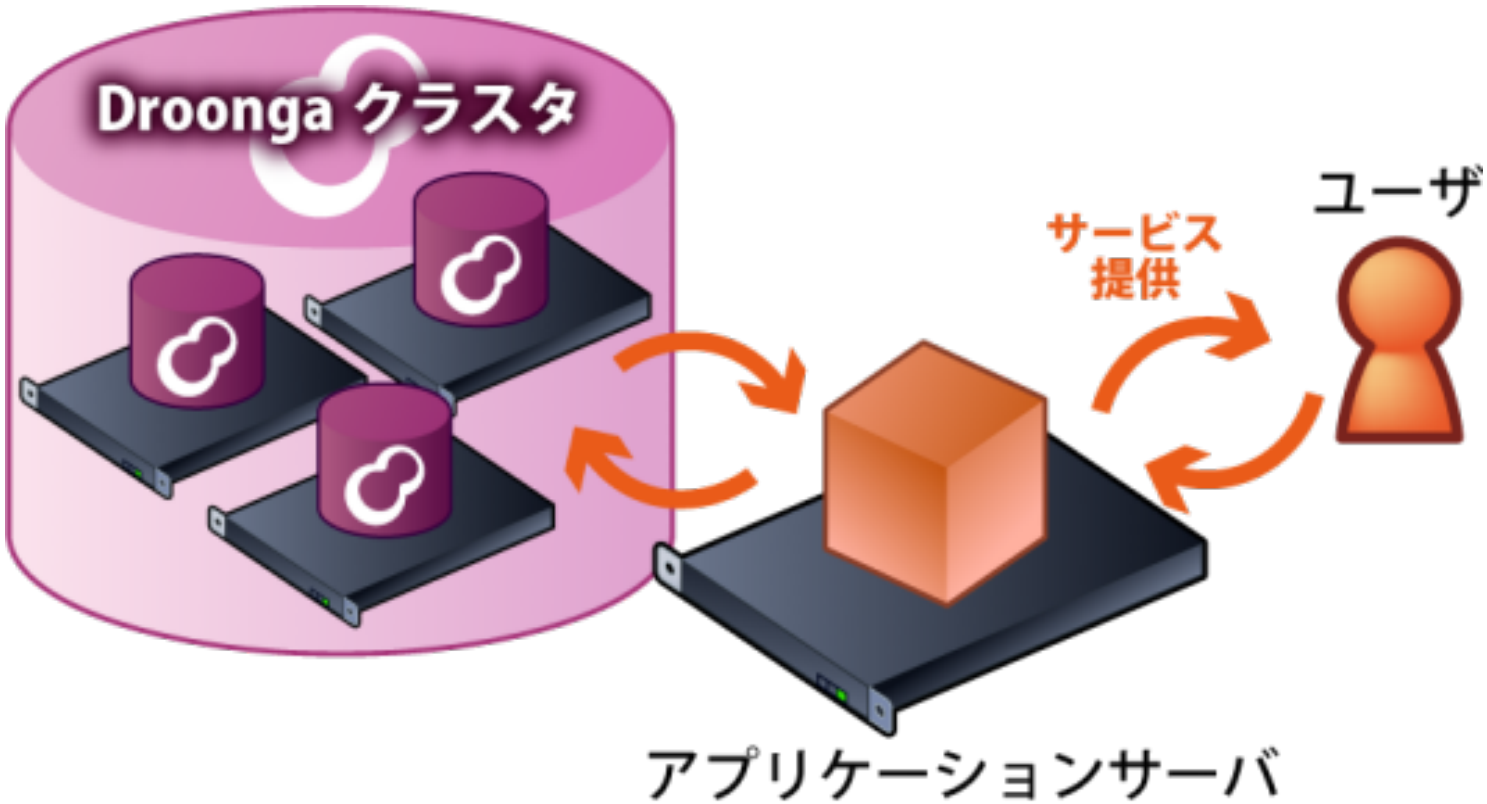
データが自動的に複製される

レプリケーション有りだと(1)



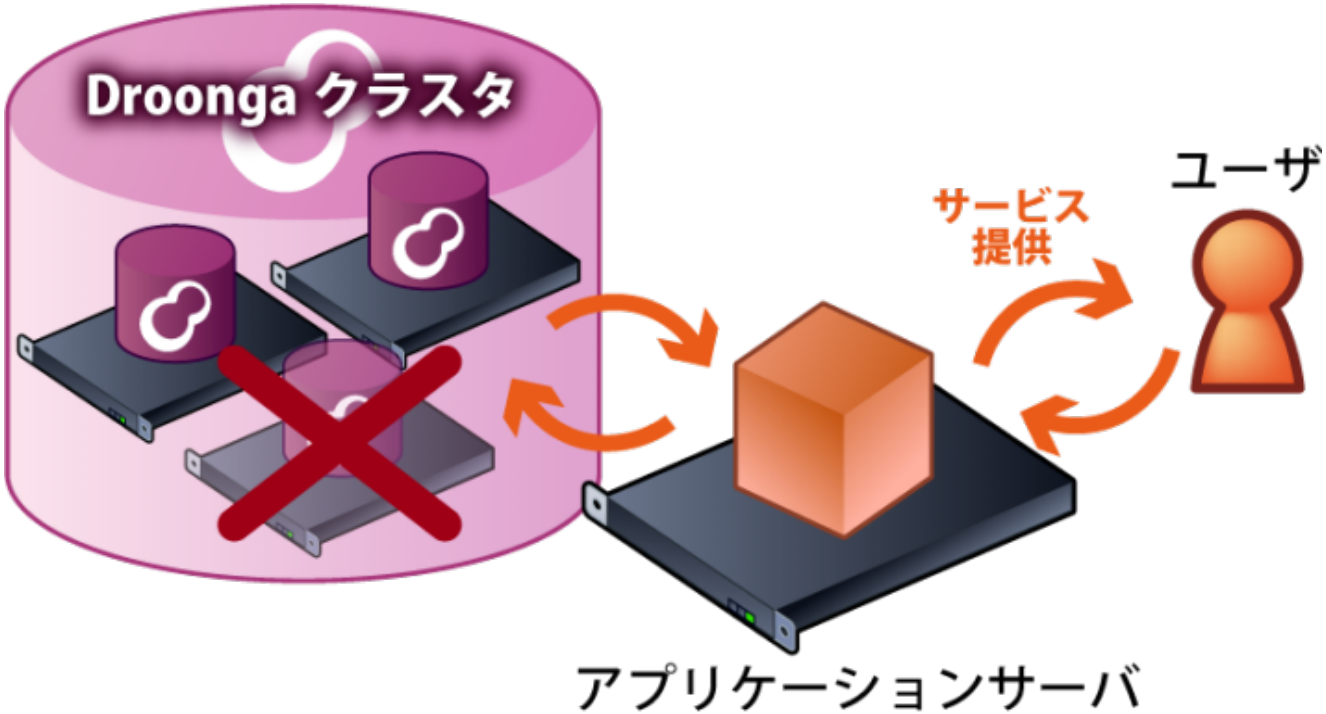
耐障害性が高くなる

レプリケーション有りだと(1)



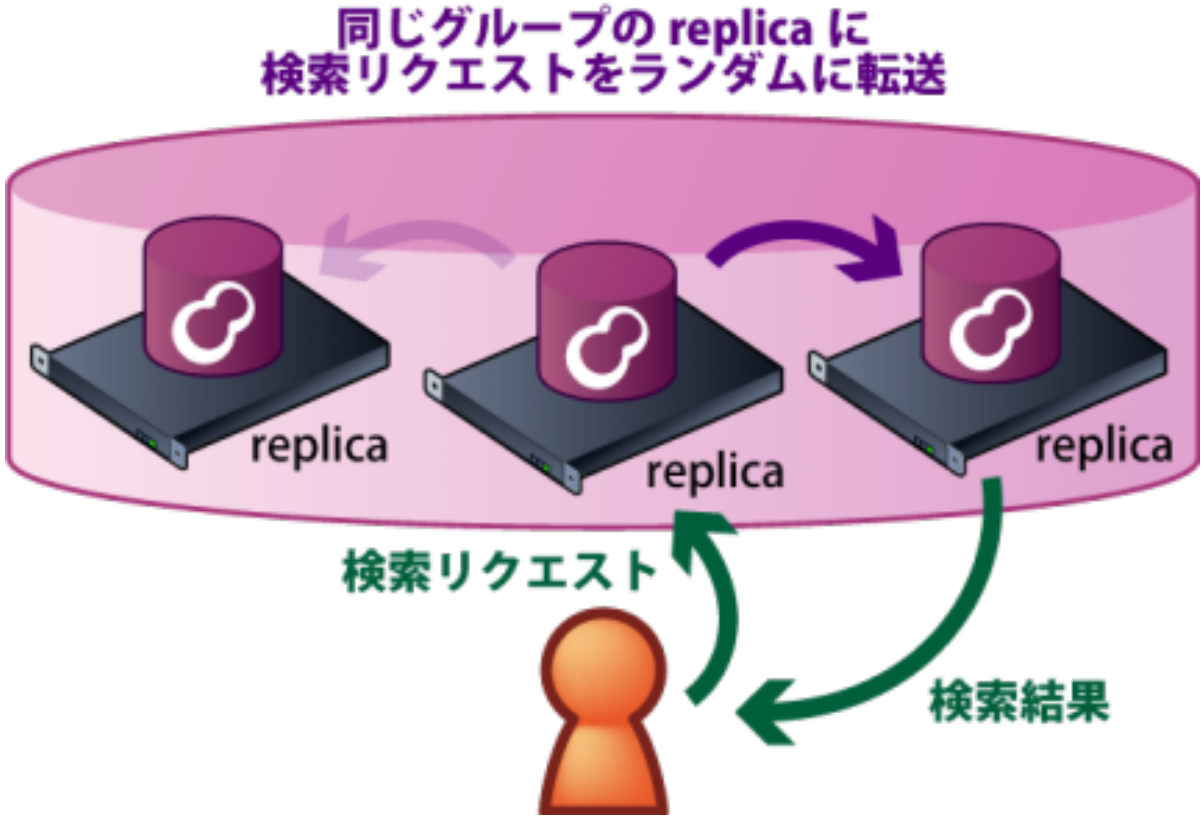
単一のサーバに依存しなくなる

レプリケーション有りだと(1)



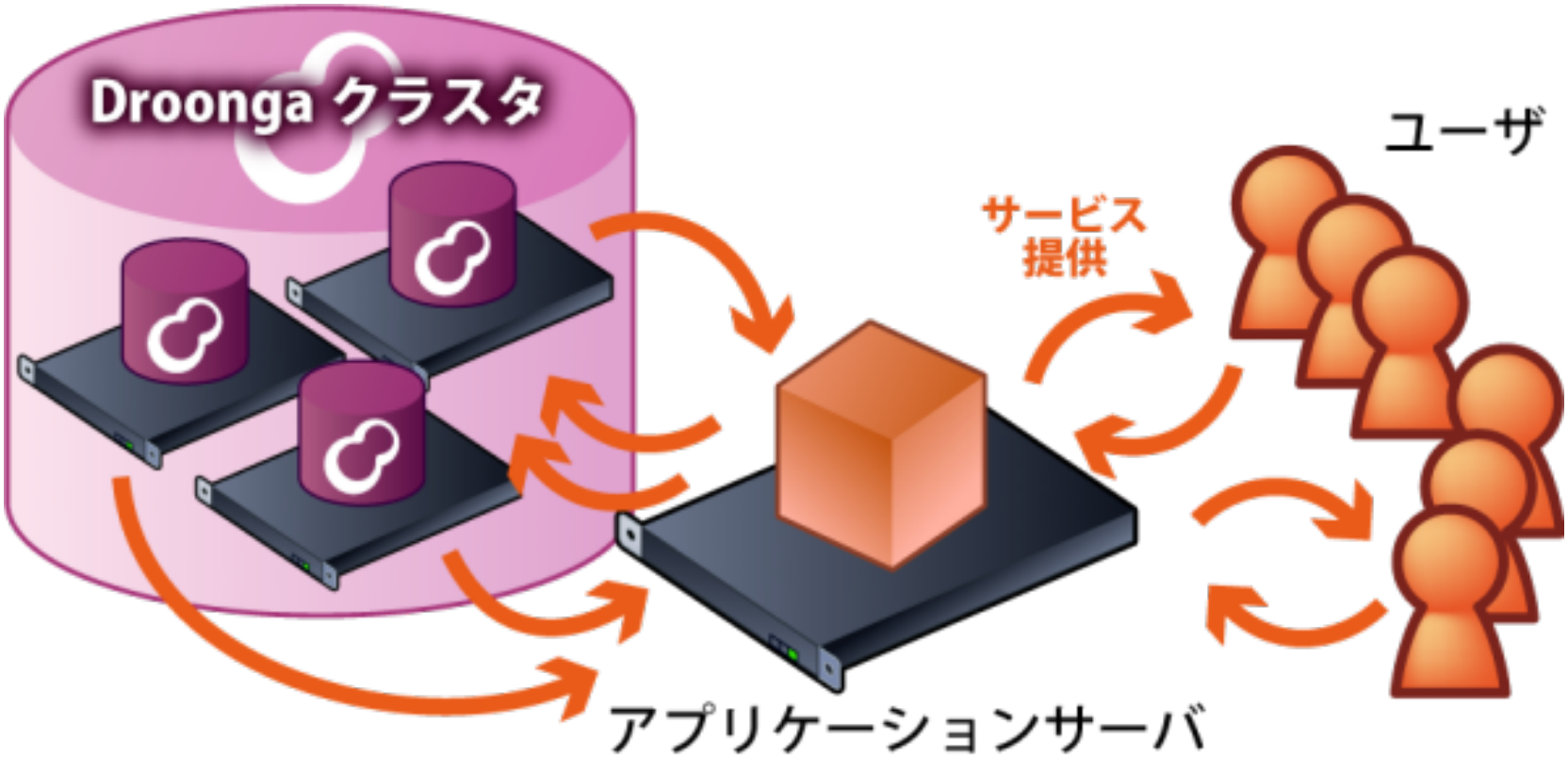
障害があってもサービスを
提供し続けられる

レプリケーション有りだと(2)



負荷が分散される

レプリケーション有りだと(2)



負荷の増大に対応しやすい

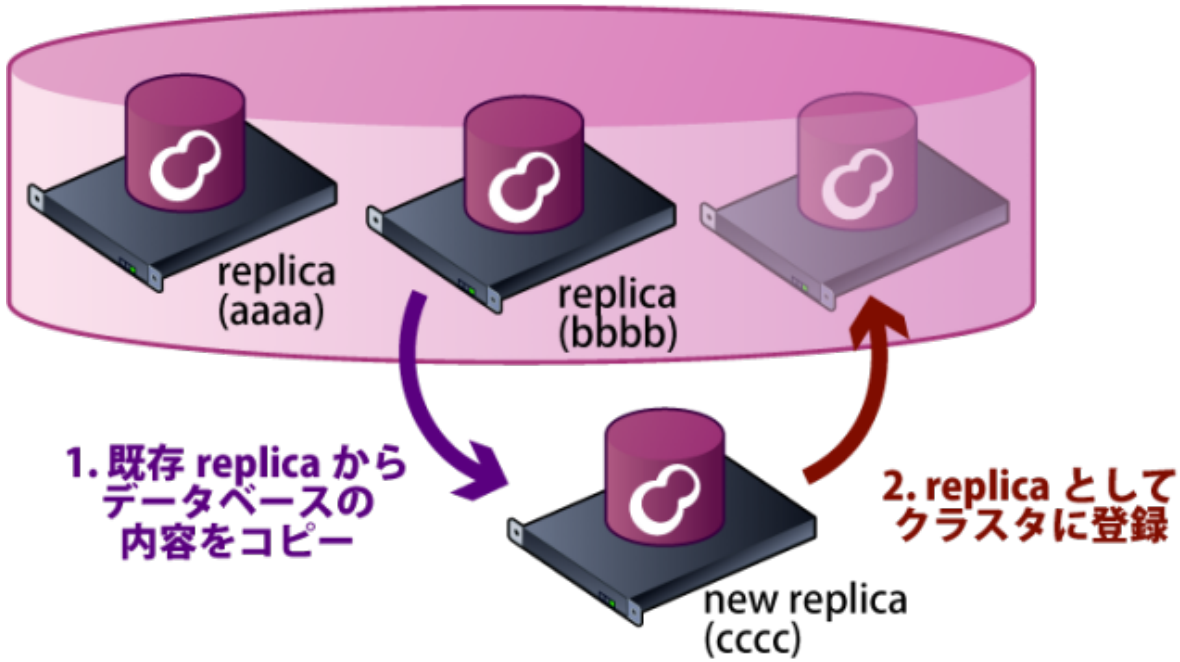
クラスタ構成の変更

付属のコマンドラインユーティリティを使用

- droonga-engine-join
- droonga-engine-unjoin

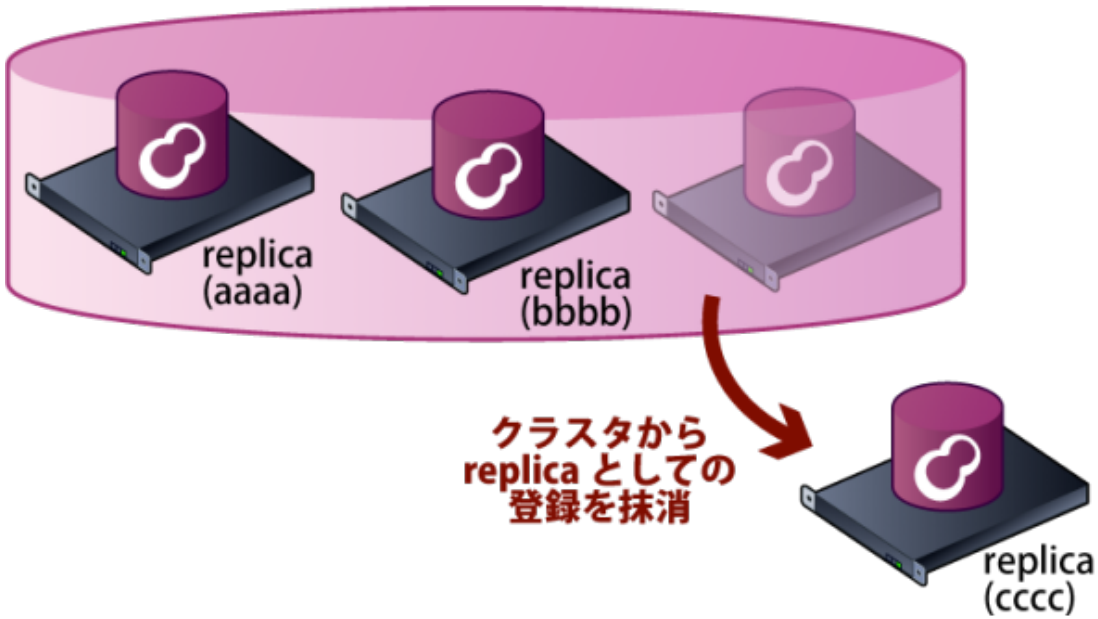
ノードの追加

```
% droonga-engine-join --host=cccc --replica-source-host=bbbb
```



ノードの切り離し

```
% droonga-engine-unjoin --host=cccc
```



デモ

- Groongaベースの
  アプリケーションを作成
- Droongaクラスタを構築
- バックエンドをDroongaに移行

（Groongaユーザ向けの、はじめてのDroonga
と同内容です）

デモ:用意する物

- サーバ2つ
  - 192.168.100.50
  - 192.168.100.51

Groonga→Droonga

今現在あるデメリット

- **レイテンシーが低下**する
  - 処理のオーバーヘッドがある
- **Groonga非互換**の部分がまだある

Groongaとの性能比較

検索対象

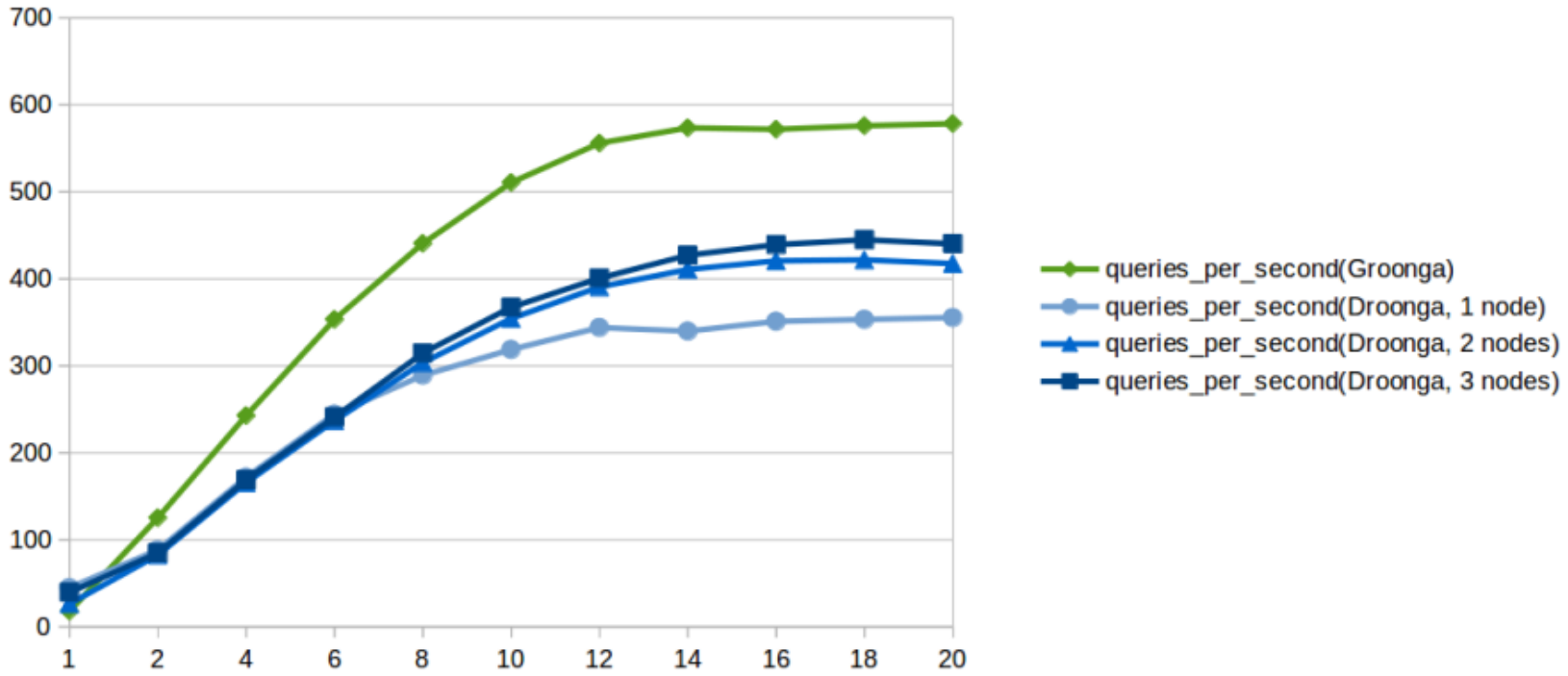
Wikipedia日本語版のページ
30万件/150万件
（データベースサイズ 1.1GiB/4.3GiB）

検索クエリ

ページのタイトル200件
（キャッシュヒット率50%）

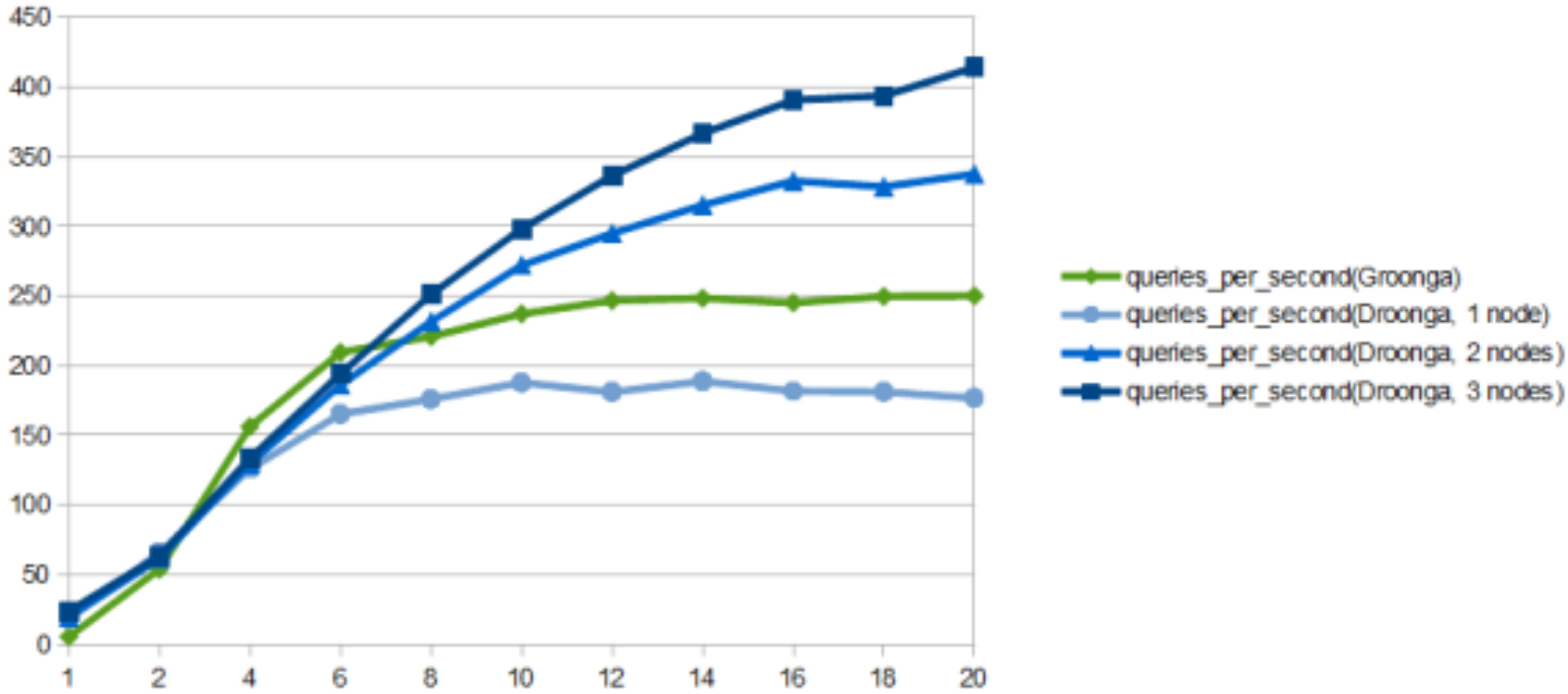
ベンチマーク取得手順

Groongaとの性能比較



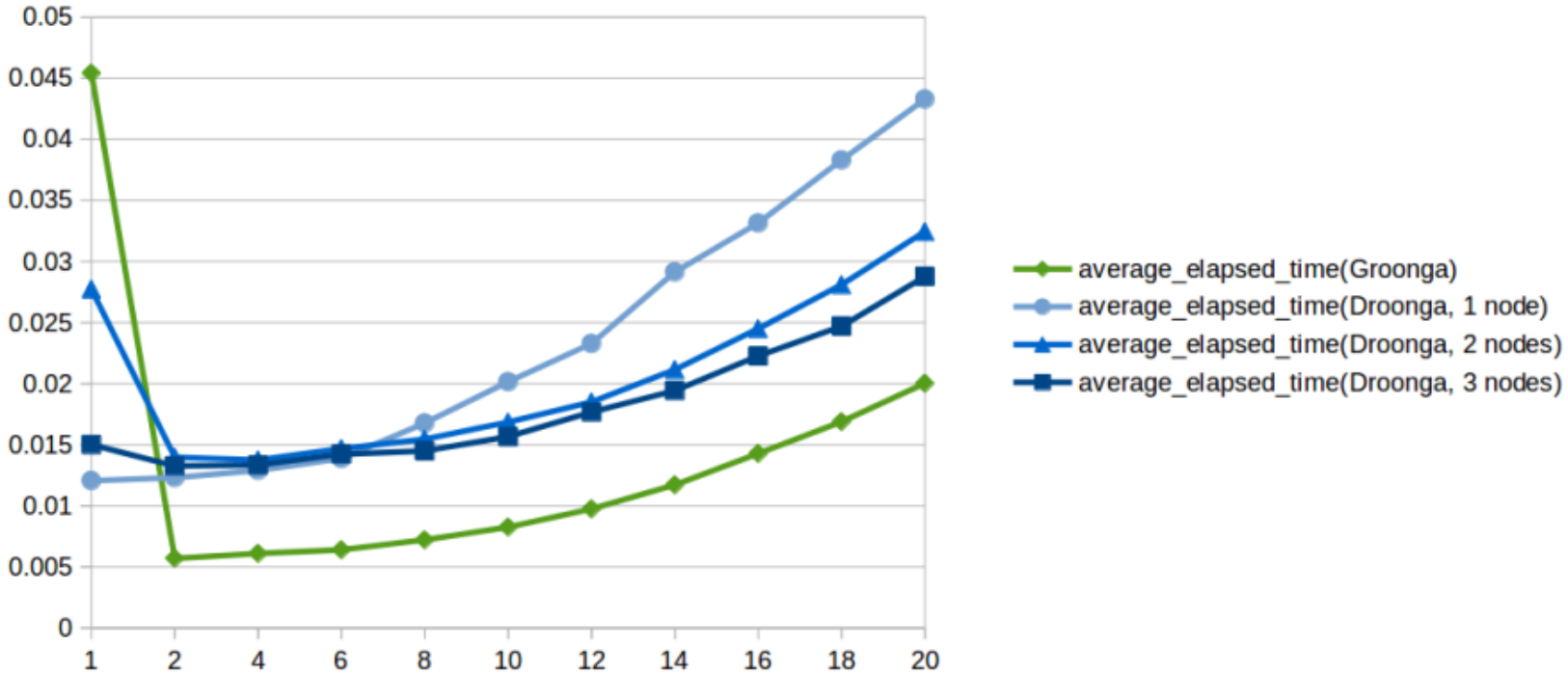
スループット（30万件）

Groongaとの性能比較



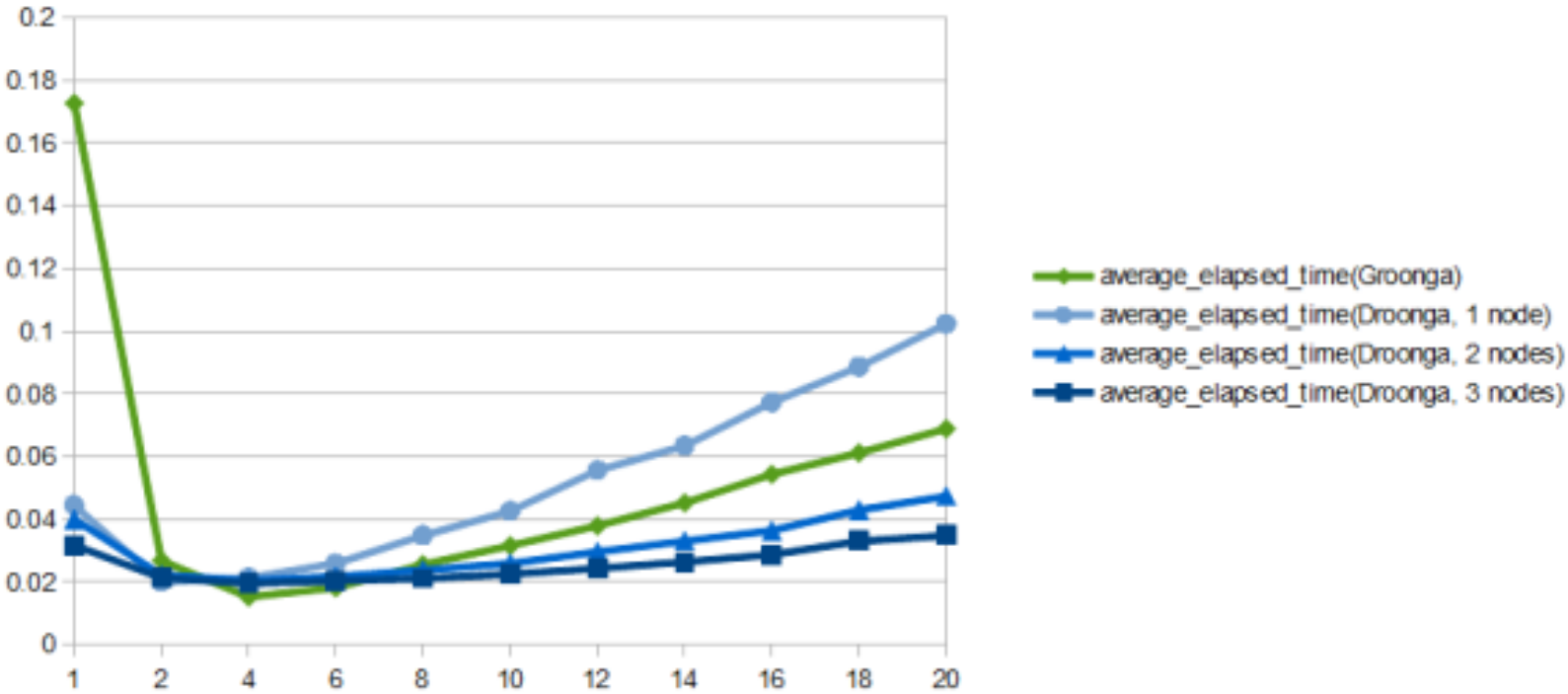
スループット（150万件）

Groongaとの性能比較



レイテンシー（30万件）

Groongaとの性能比較



レイテンシー（150万件）

それぞれの傾向

Groonga

- ✓ サーバ1台の処理能力では有利
- ✓ 負荷が増えるとスループットが頭打ちになる

Droonga

- ✓ サーバ1台の処理能力では不利
- ✓ ノード追加でスループットの上限が増える

傾向の分析

検索処理そのものが軽い時

- ✓ オーバーヘッドの影響が相対的に大
- ✓ Groongaの方が有利

検索処理そのものが重い時

- ✓ オーバーヘッドの影響が相対的に小
- ✓ Droongaの方が有利

Droongaの方が有利な場面

- データベースが大きい
- 重いクエリが多い
- …

今分かっている遅くなる理由

- ドリルダウンがあると遅くなる
- レスポンスのサイズが大きくなると遅くなる
- クラスタ構成に合わせた処理の最適化が不十分

現時点での互換性(概要)

- スキーマ変更系
- load, delete
- select

それ以外は未対応(今後の課題)

現時点での互換性（詳細）

table操作系

- table_create
- table_remove
- table_list

現時点での互換性（詳細）

column操作系

- column_create
- column_remove
- column_rename
- column_list

現時点での互換性（詳細）

データ更新系

- load
  - ただし、以下は未対応
    --ifexists, --input_type
- delete

現時点での互換性（詳細）

検索系

- select
  - ただし、以下は未対応
    --scorer, --cache,
    --match_escalation_threshold,
    --query_expansion,
    --query_flags, --query_expander,
    --adjuster

とはいえ

Groongaの利用状況次第では
今すぐにでもDroongaに移行できる！

改善にご協力を!

様々な条件でのベンチマーク結果

- できればGroongaでの
  ベンチマーク結果も添えて
- 開発チームで認識できていない
  ボトルネックを明らかにしたい
- ベンチマーク取得手順公開中

改善にご協力を!

実際のWebアプリケーションからの
使い勝手レポート

- 互換性向上の作業の
  優先順位付けの参考にしたい

改善にご協力を!

フィードバックは
droonga-engineのissue tracker
もしくは
groonga-dev ML
までお寄せ下さい