分散KVSにおけるメニーコアを活用したプロキシによるクエリ集約及び排他制御

◯三輪 竜也[1]　川浪 大知[1,2]　川島 龍太[1]　松尾 啓志[1]

[1] 名古屋工業大学
[2] 現在，株式会社ボスコ・テクノロジーズ

研究背景

- 分散キーバリューストア (KVS)
  - キーとバリューからなる単純なデータ構造
  - 複数のノードでデータを分散管理

通信処理における問題点

排他制御における問題点

プロキシを用いたクエリ集約

クエリ集約の有効性を確認†

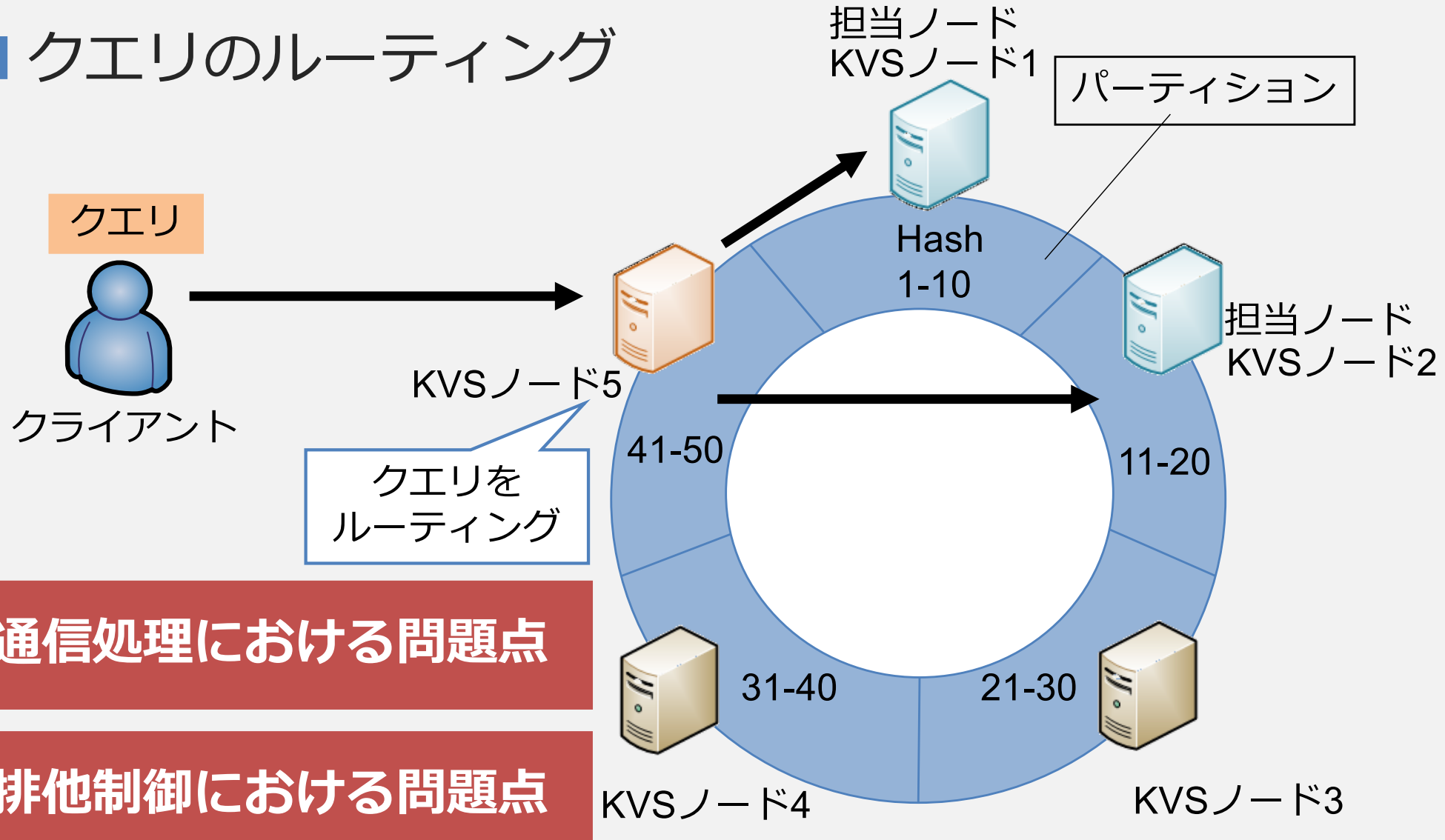
提案手法

- **プロキシの高速化**
- **プロキシによる排他制御**

† Daichi Kawanami et al. “A Proxy-Based Query Aggregation Method for Distributed Key-Value Stores" , Proc. 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2018

リングアーキテクチャ

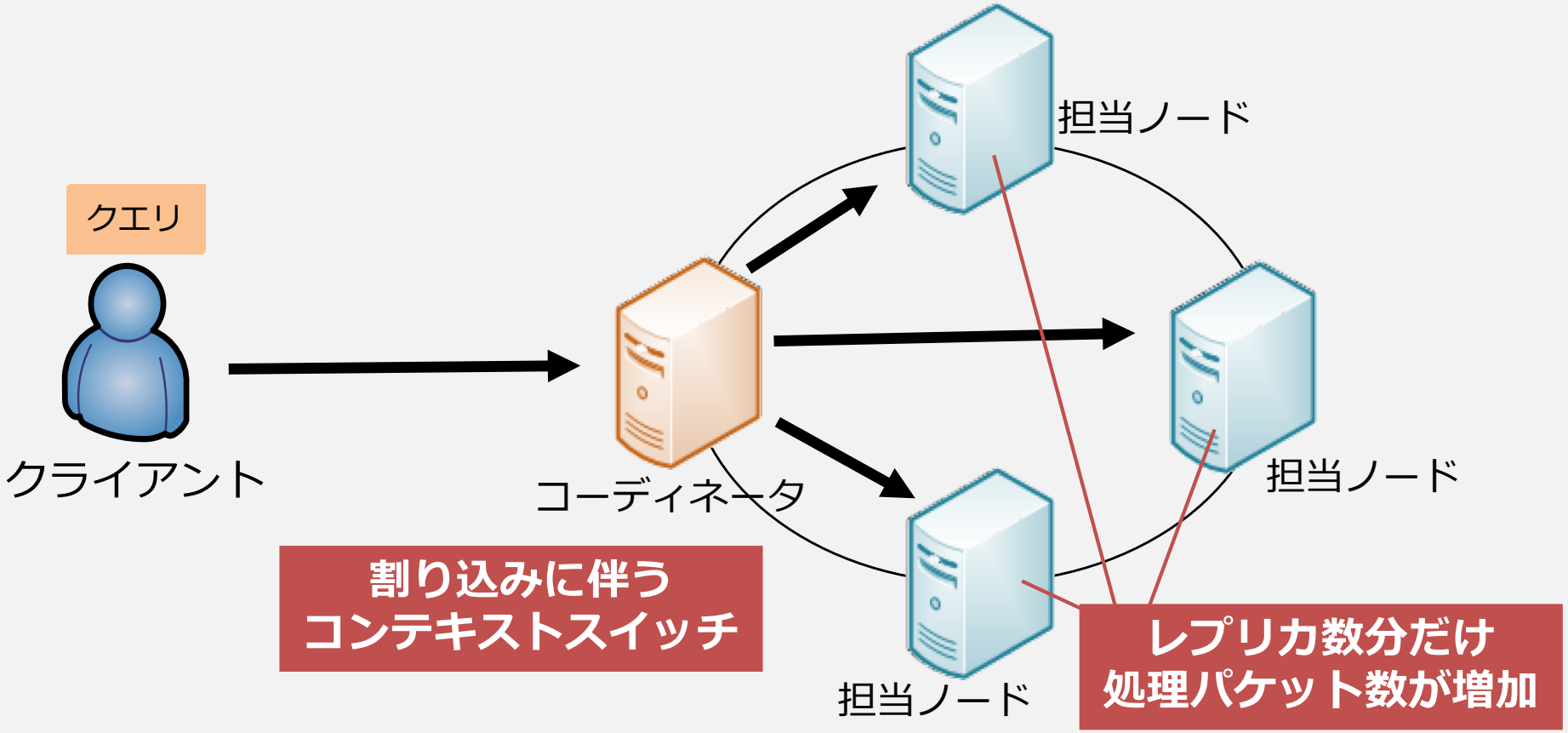
- コンシステントハッシングによってデータを分散
- クエリのルーティング



通信処理における問題点

排他制御における問題点

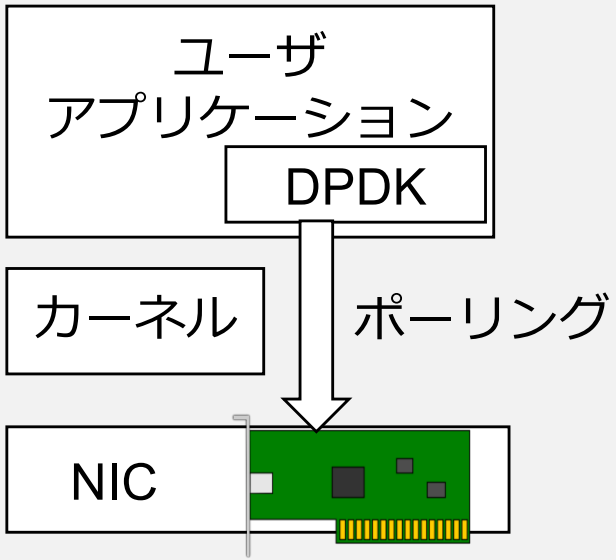
通信処理における問題点



**クエリ転送で発生するパケット数を削減し、
ネットワークの処理負荷を軽減することが必要**

パケット処理の高速化

- Data Plane Development Kit (DPDK)
  - ユーザ空間からNICをポーリングすることで **コンテキストスイッチを回避**



- ScyllaDB
  - ネットワークスタックにDPDKを使用
  - Cassandra互換な分散KVS
  - Cassandraの10倍のスループットを達成†1

DPDKを用いることで分散KVSの性能が向上

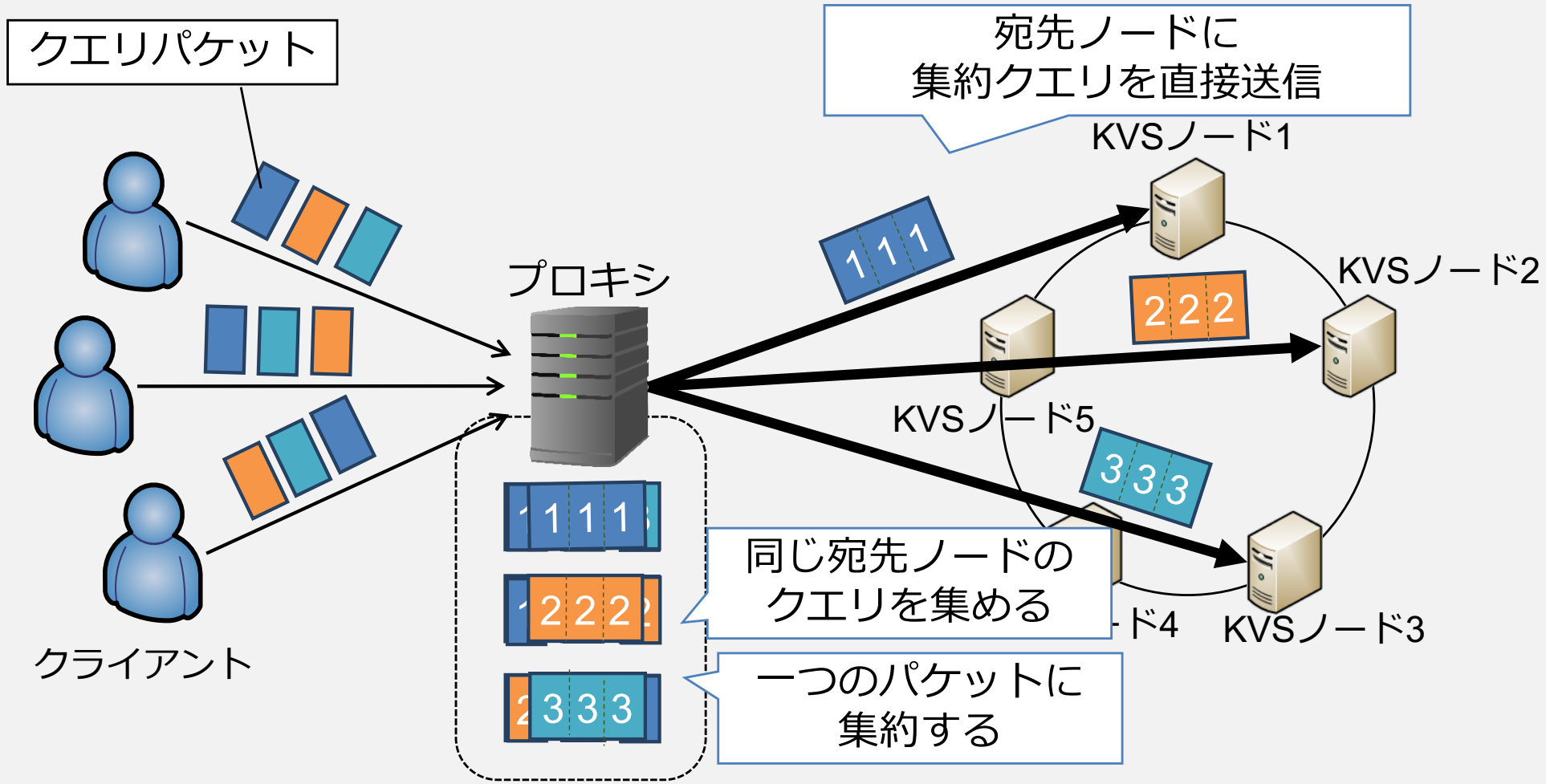
DPDKを用いるだけではショートパケットを ワイヤレートで処理できない†2

†1 https://www.scylladb.com/product/benchmarks/

†2 R. Kawashima et al., “Evaluation of Forwarding Efficiency in NFV-nodes toward Predictable Service Chain Performance,” IEEE Trans. on Network and Service Management, 2017.

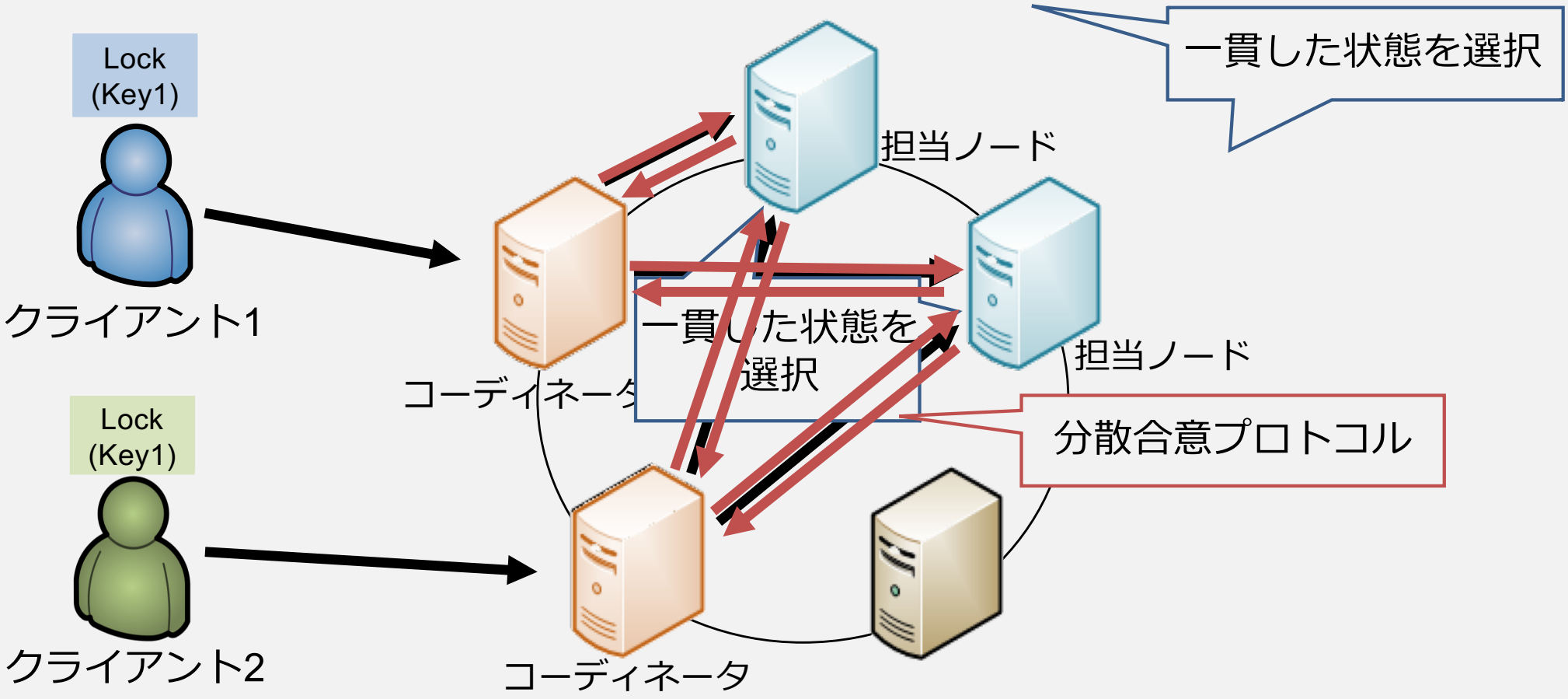
プロキシによるクエリ集約†

- クエリ集約により**パケット数を削減**



† Daichi Kawanami et al. “A Proxy-Based Query Aggregation Method for Distributed Key-Value Stores" , Proc. 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2018

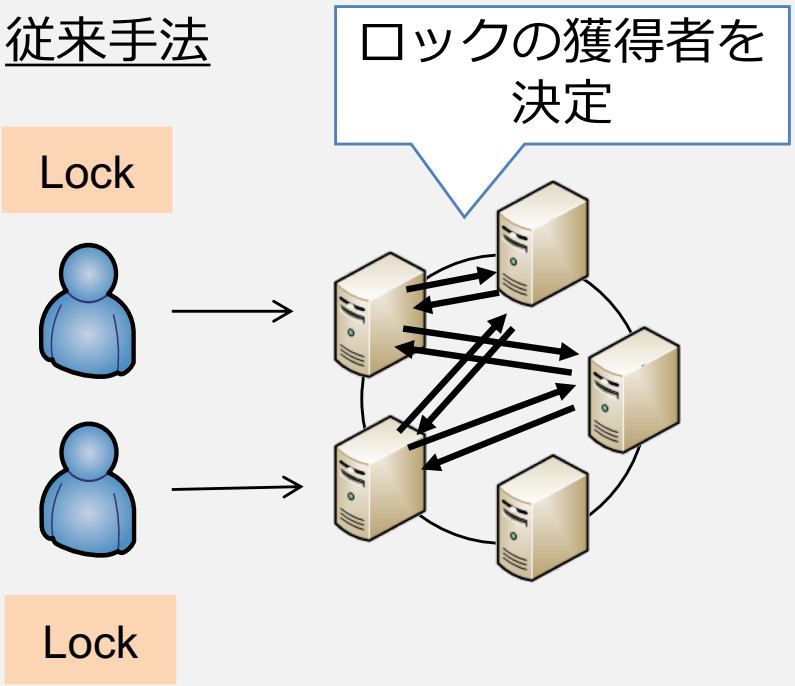
排他制御における問題点

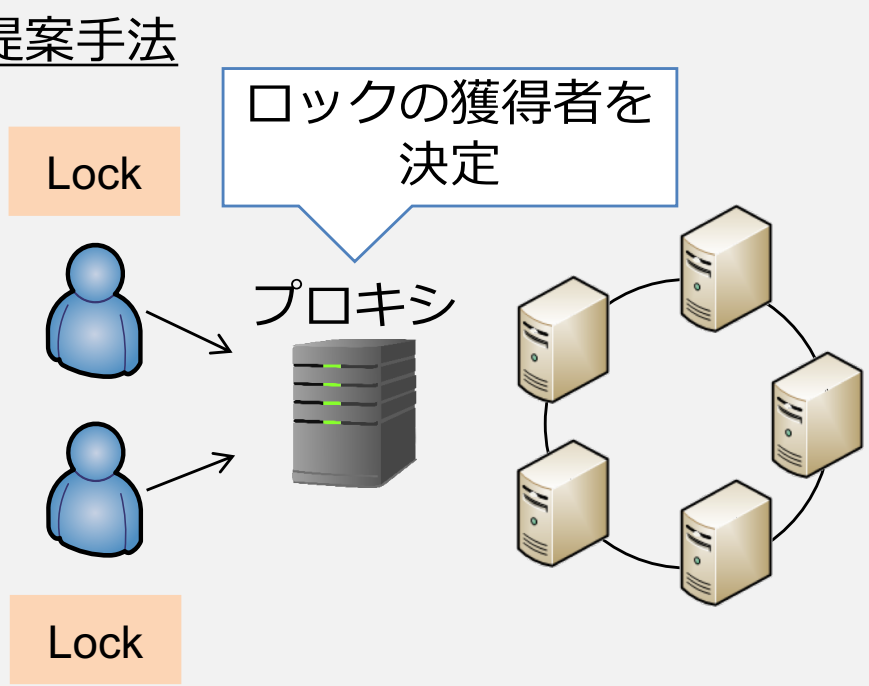


排他制御のための通信が増加

プロキシによる排他制御

- プロキシ上で排他制御することで一貫性制御の通信を削減

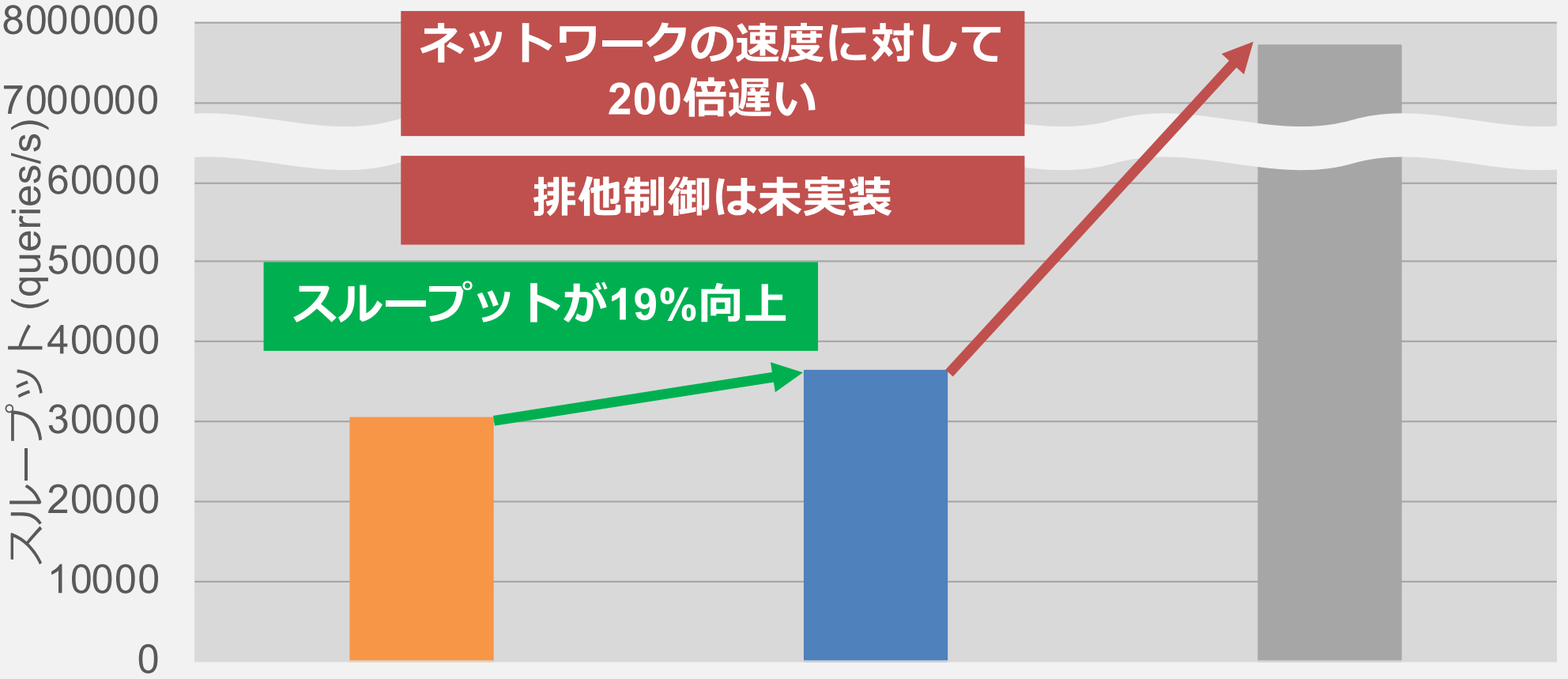




KVSノード間で一貫性制御が必要

KVSノード間で一貫性制御が不要

クエリ集約の有効性

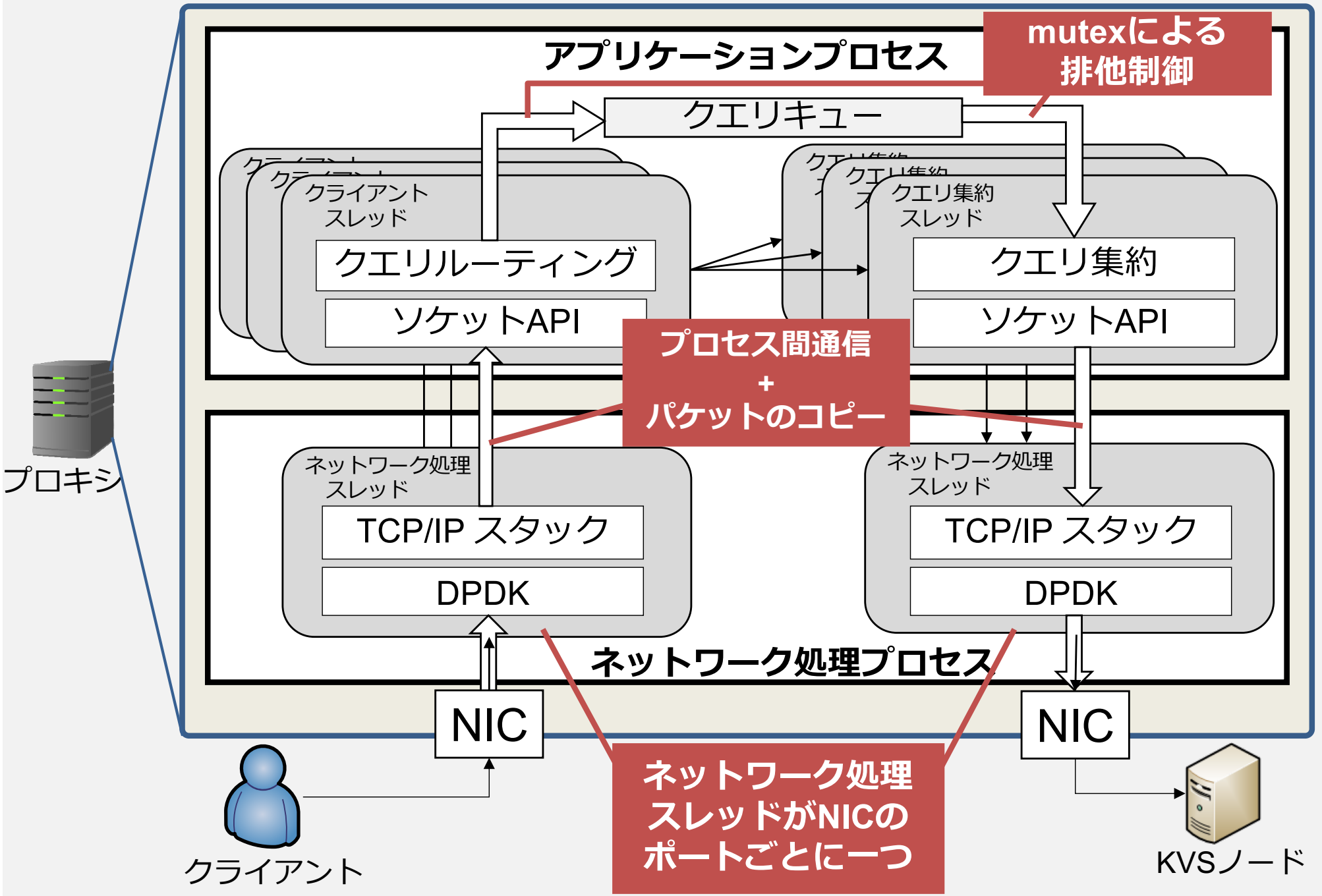


提案手法

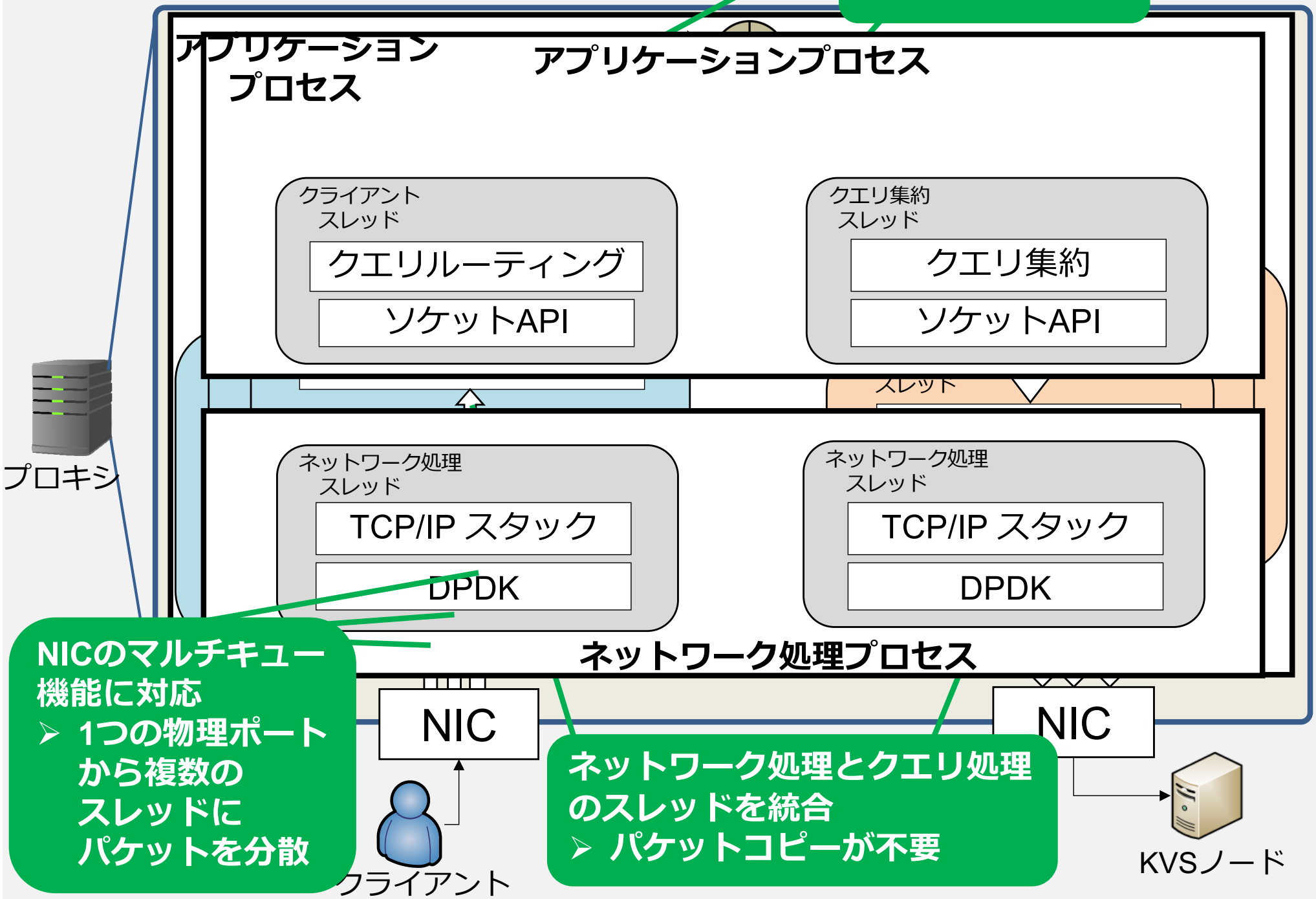
- プロキシの高速化
- プロキシによる排他制御

れた場合

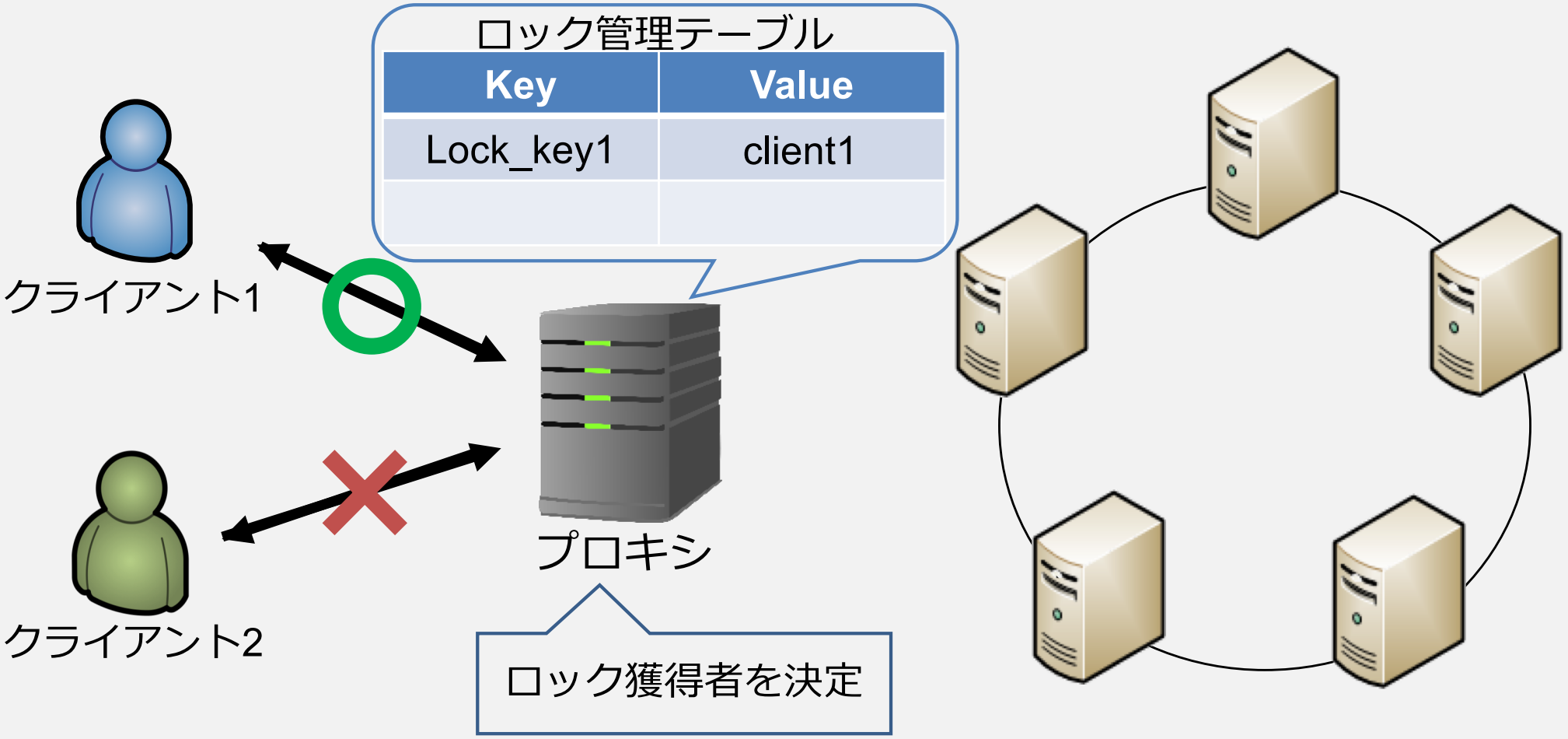
初期のプロキシの実装における問題点



プロキシの高速化実装

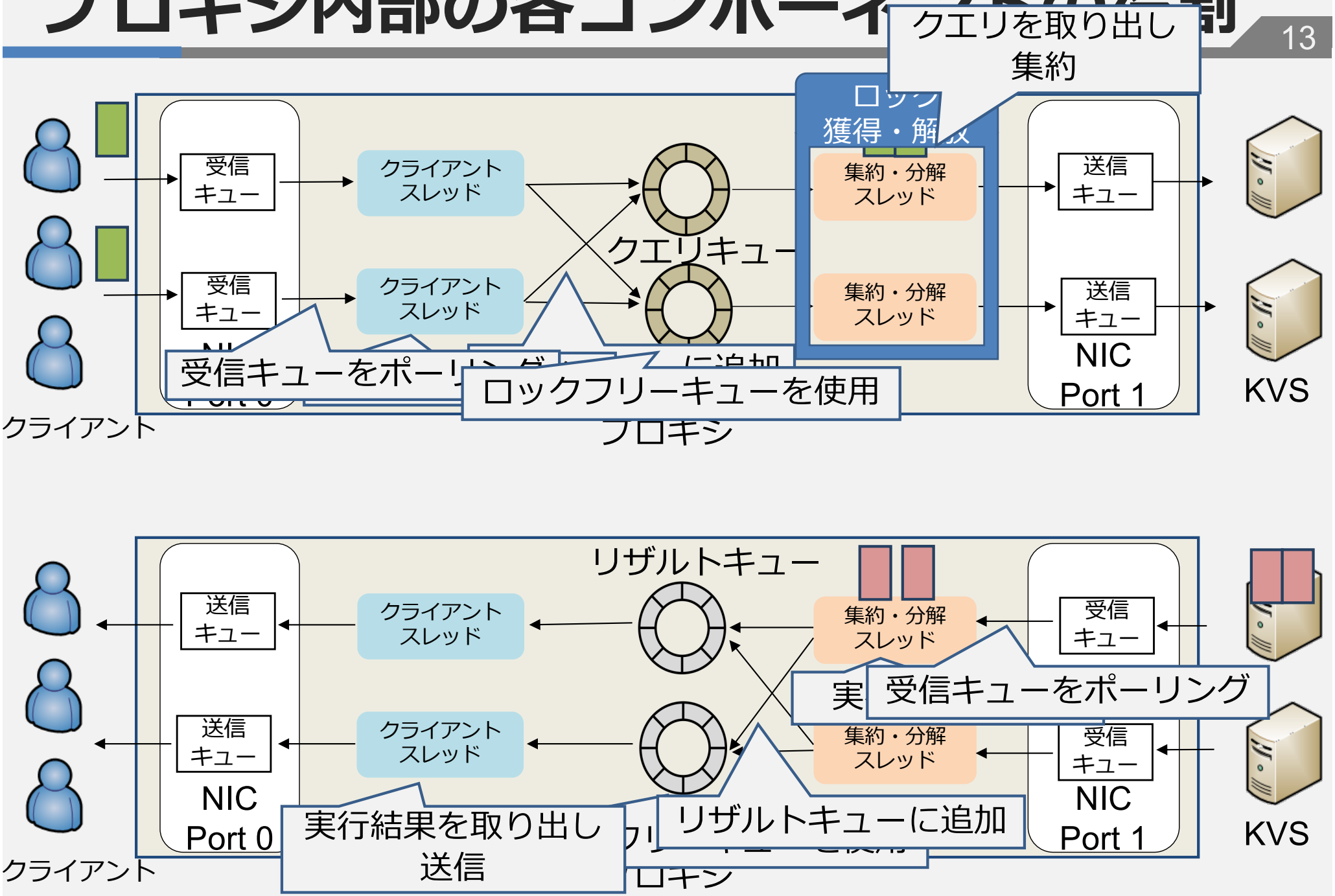

CAS命令を用いた
排他制御
アプリケーション
プロセス
アプリケーションプロセス
クライアント
スレッド
クエリルーティング
ソケットAPI
クエリ集約
スレッド
クエリ集約
ソケットAPI
プロキシ
ネットワーク処理
スレッド
TCP/IP スタック
DPDK
ネットワーク処理
スレッド
TCP/IP スタック
DPDK
ネットワーク処理プロセス
NIC
NIC
NICのマルチキュー
機能に対応
➢ 1つの物理ポート
から複数の
スレッドに
パケットを分散
ネットワーク処理とクエリ処理
のスレッドを統合
➢ パケットコピーが不要
クライアント
KVSノード

プロキシによる排他制御



排他制御のための通信を削減

プロキシ内部の各コンポーネントの役割

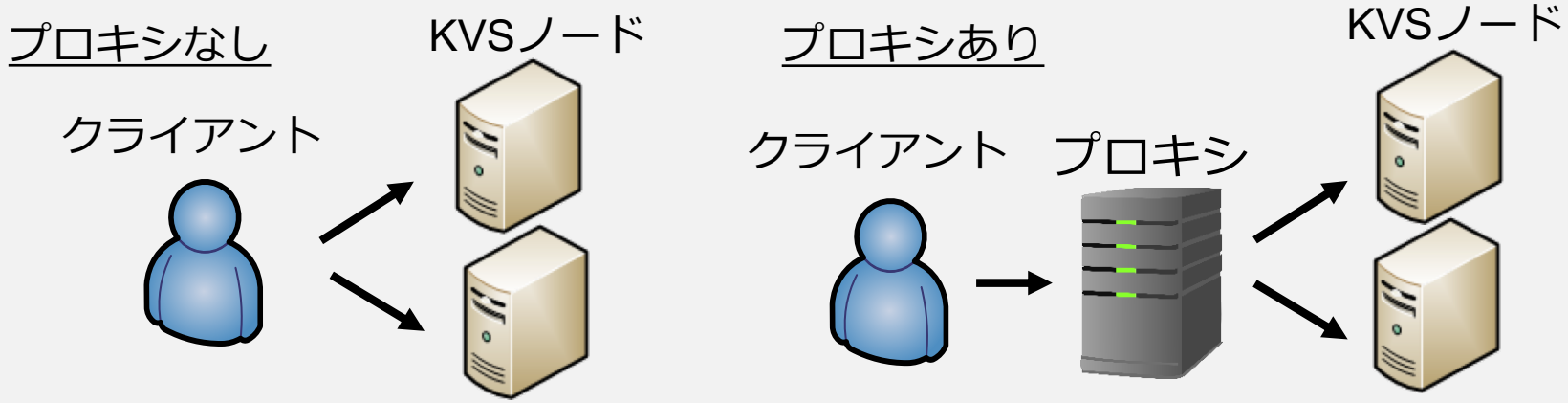


プロキシの性能評価における評価環境

- KVSノード
  - DPDKを利用
  - クエリの受信と結果の送信処理
  - 少数ノードで実験を行うため、データの永続化は行わない
- クライアント
  - DPDKを利用
  - 2種類の動作

プロキシなし：直接KVSノードに送信 (1 hop)

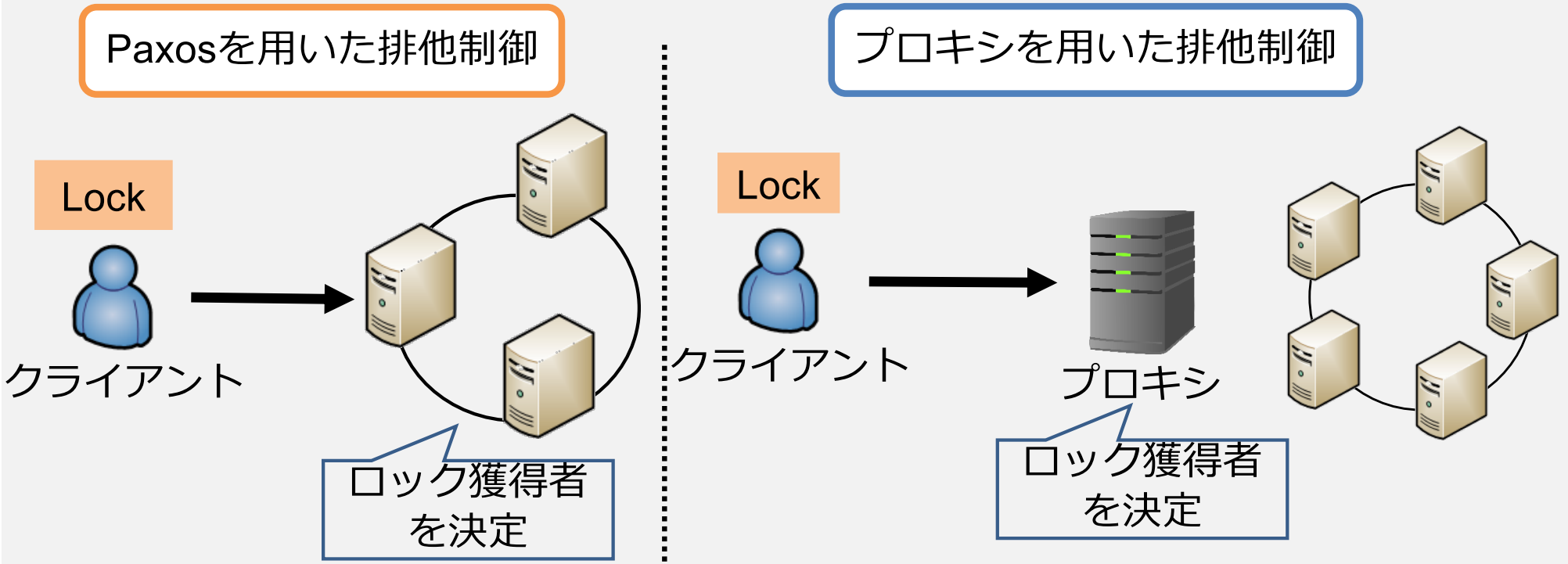
プロキシあり：プロキシにクエリを送信（2 hop）



ロック獲得評価における評価環境

- KVSノード
  - Cassandraを使用

- 2通りの環境でロック獲得レイテンシを評価



評価環境

マシンスペック

マシン性能 (プロキシ)	
CPU	Core i9-7980XE 2.6 GHz, 18 cores / 36 threads
RAM	64 GB
NIC	Intel X550T (10 GbE, dual port)
OS	Ubuntu 16.04

マシン性能 (クライアント,KVSノード,Cassandraノード)	
CPU	Core i5-4460 3.2 GHz, 4 cores / 4 threads
RAM	16 GB
NIC	Intel X540-T2(10 GbE, dual port)
OS	Ubuntu 16.04

プロキシ性能評価

マシン台数

- クライアント:3台
- KVSノード:5台
- プロキシ:1台

ワークロード

- キーバリューサイズ 10bytes
- 書き込みクエリ
- 10万種類のキーを一様に生成

ロック獲得評価

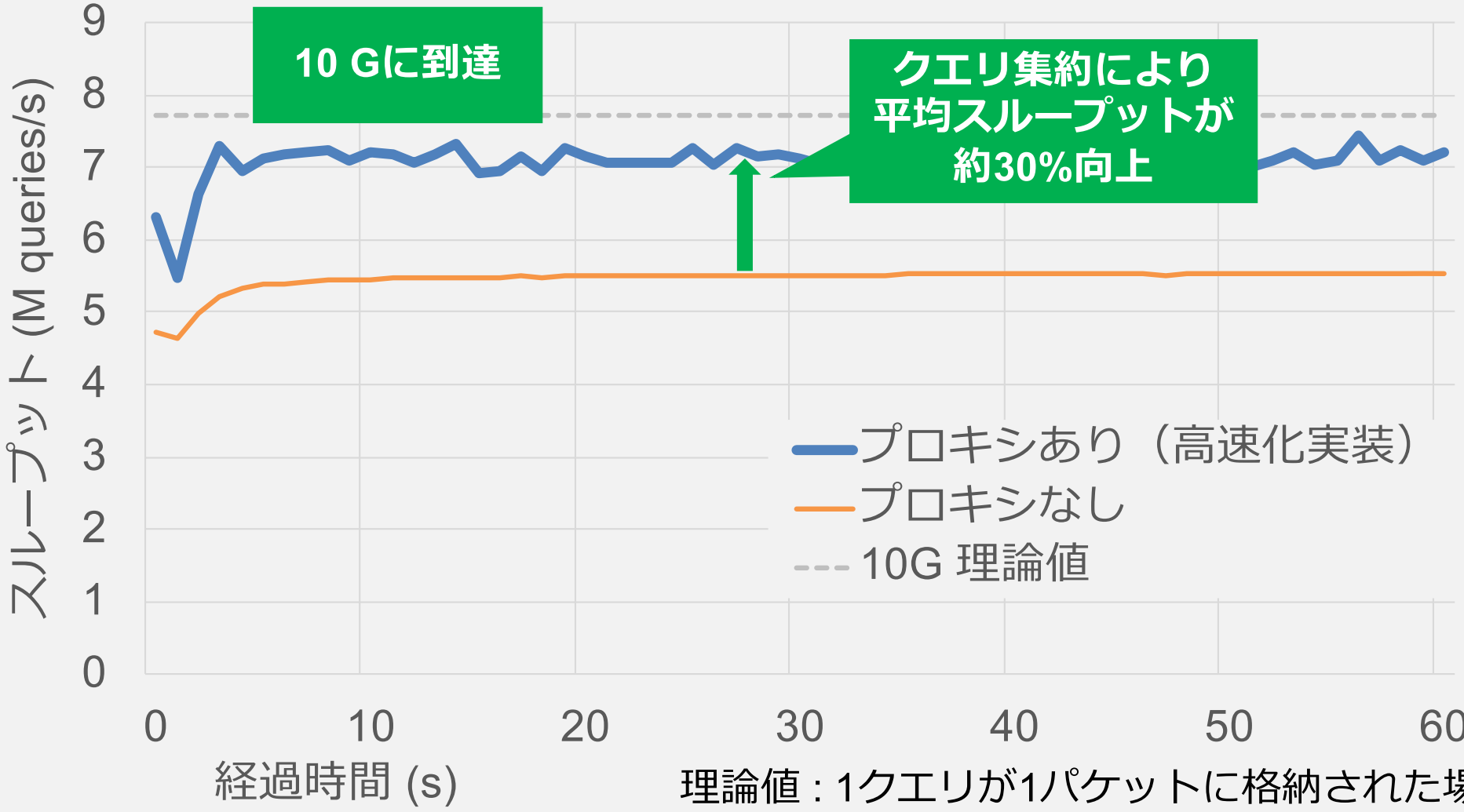
マシン台数

- クライアント:1台
- Cassandraノード
  - 5台(プロキシあり)
  - 3～9台(プロキシなし)
- プロキシ:1台

ワークロード

- キーバリューサイズ 10bytes
- ロック獲得クエリ
- 100万種類のキーを一様に生成

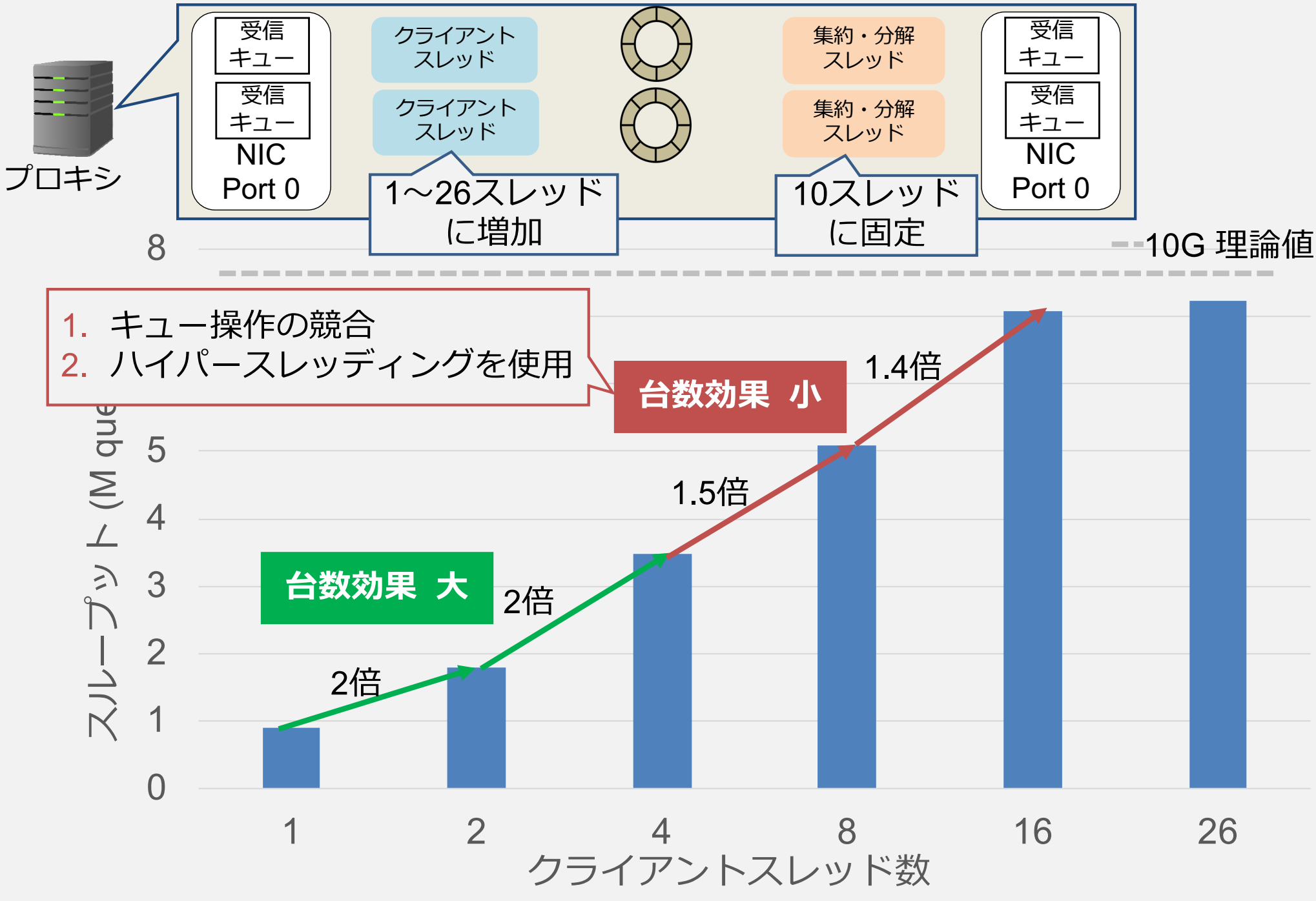
プロキシの性能評価



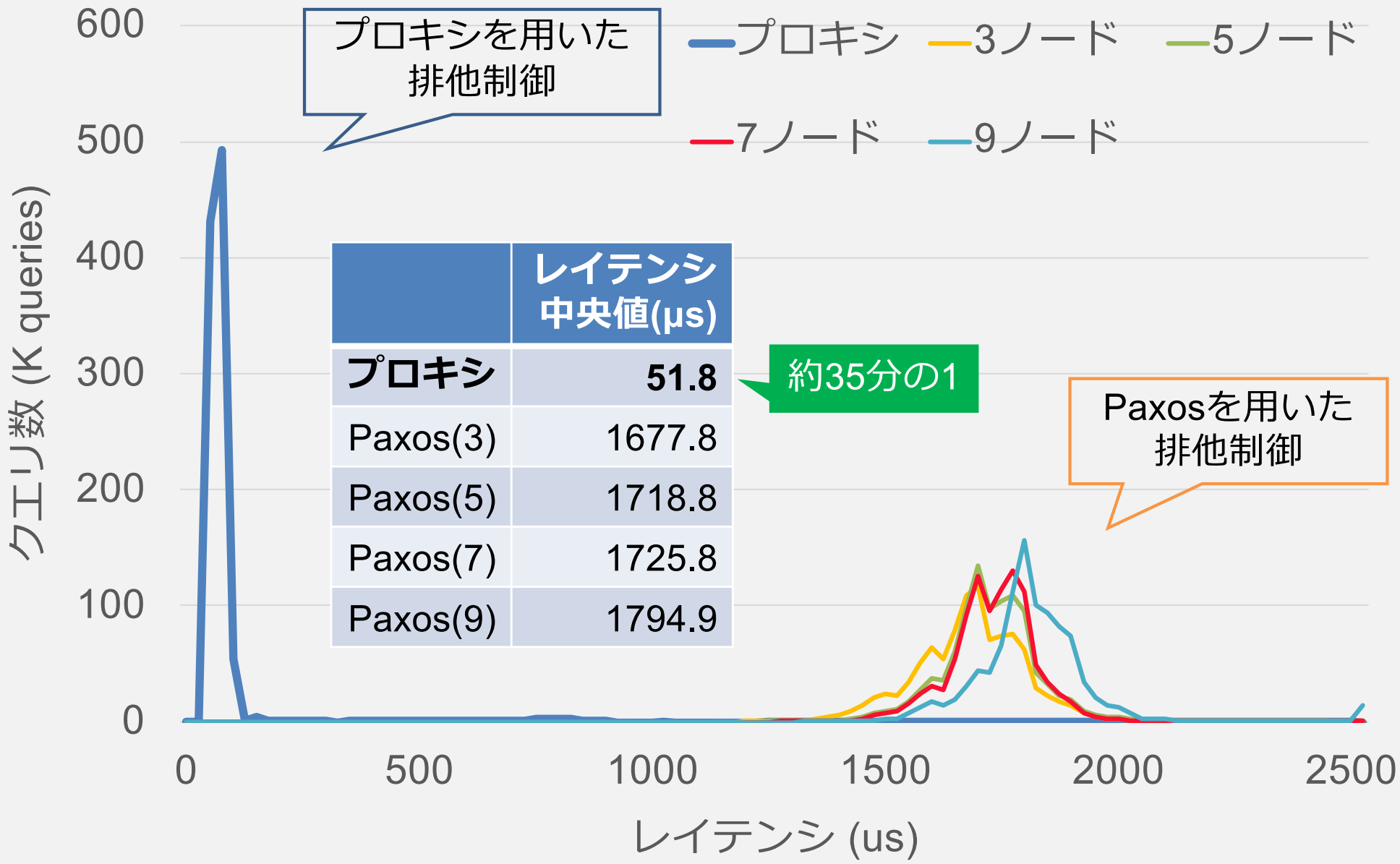
理論値：1クエリが1パケットに格納された場合
1sに送信可能なクエリ数

レイテンシ	最小値(ms)	中央値(ms)	平均値(ms)	最大値(ms)
プロキシあり	0.051	217	412	9995
プロキシなし	0.035	0.48	1.6	2328

スケーラビリティの評価



ロック獲得レイテンシの評価



	レイテンシ 中央値(μs)
プロキシ	**51.8**
Paxos(3)	1677.8
Paxos(5)	1718.8
Paxos(7)	1725.8
Paxos(9)	1794.9

まとめと今後の課題

まとめ

- クエリ集約を行うプロキシの性能を改善
  - マルチキューへの対応
  - プロセス統合による、プロセス間通信の削減
  - CAS命令によるスレッド間の通信コストの削減
  - 10 Gbpsのスループットを持つプロキシを実現
- プロキシに排他制御機能を追加
  - KVS間での排他制御のための通信を削減

今後の課題

- トランザクション処理機能の実現
- プロキシの単一故障点解消