【奨励講演】
メッセージバッファの再設計による実CNFのキャッシュ効率化

○山田 歩人[†]　　川島 龍太[†]

中山 裕貴[††]　　林 經正[††]　　松尾 啓志[†]

[†]　名古屋工業大学大学院

[††]　株式会社ボスコ・テクノロジーズ

研究の概要

目的

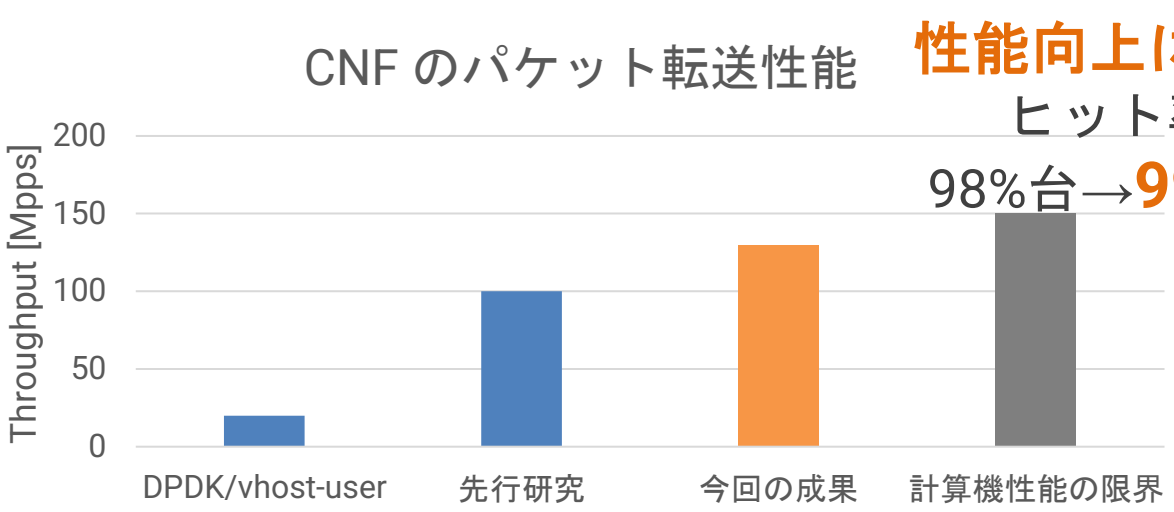
実サービスとして展開できる形で，CNFのパケット転送性能を最大化する

アイデア

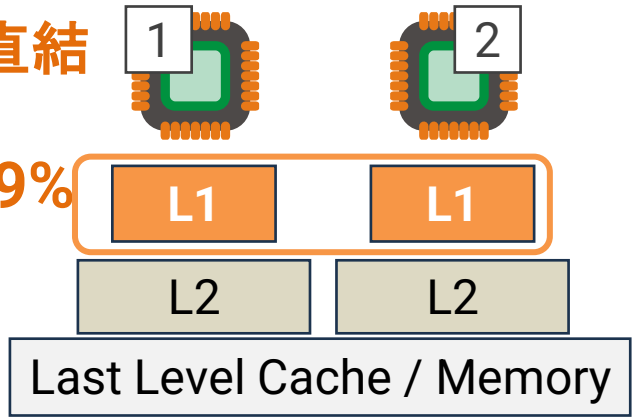
多量のパケットを一度に処理する仕組みを導入しながらも，
省メモリなメッセージバッファ構成

結果

L1キャッシュヒット率を極限まで高め，計算機性能の上限との差を縮めた

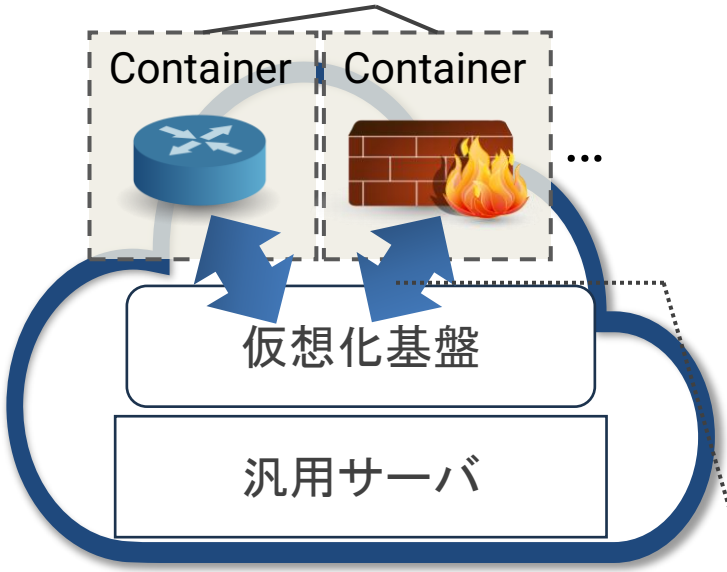


性能向上に直結
ヒット率:
98%台→**99.9%**



研究背景

完全仮想化クラウドネイティブNW
Cloud-native Network Functions



- ✓ 迅速なデプロイ
- ✓ 高スケーラビリティ

NW の世代更新に向けて CNF の性能向上が急務

- 6G の要求性能:
  デバイスあたり100 Gbps≒150 Mpps以上†
- 現状の性能:
  DPDK/vhost-user でフローあたり20 Mpps

ソフトウェアの NW I/O 性能は頭打ち

- × 一般的な実装テクニック

ボトルネック: **仮想 NW I/O**

仮想 NW I/O の性能向上

次世代NWにおいて完全仮想化の適用範囲が広まる

† Nokia Bell Labs, “Communications in the 6G era”, 2020

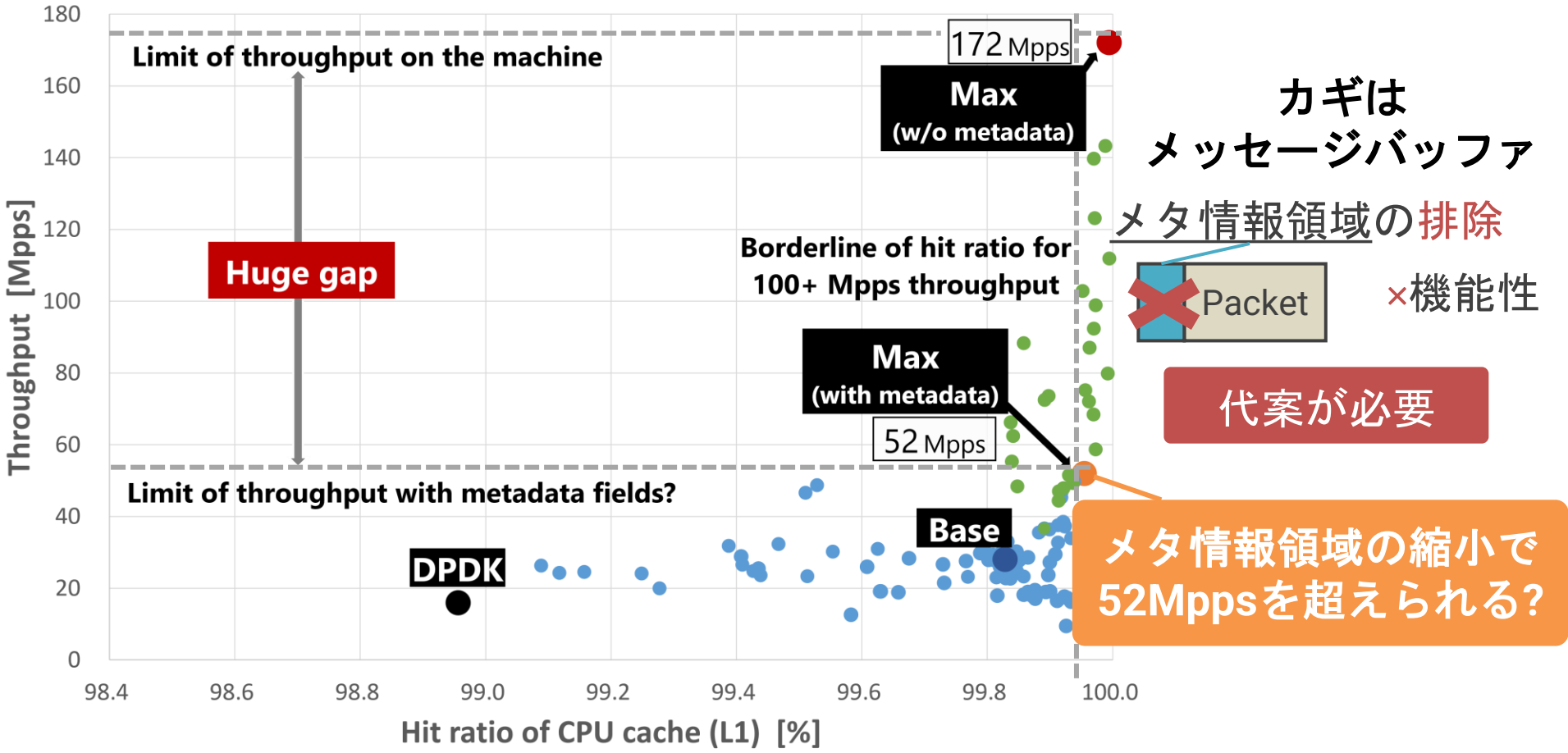
先行研究 1†

CPUキャッシュの利用効率向上が必要 ?††

総当たり的に 性能評価
141種類の実装・設定

ベース構成
（DPDKと同等）

- データ構造
- 実装方式
- 静的パラメータ



カギは
メッセージバッファ

メタ情報領域の排除

Packet

×機能性

代案が必要

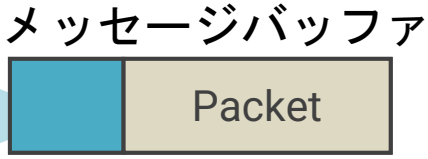
メタ情報領域の縮小で
52Mppsを超えられる?

先行研究 2† (1月NS研の発表内容)

メタ情報領域を縮小する手法を提案

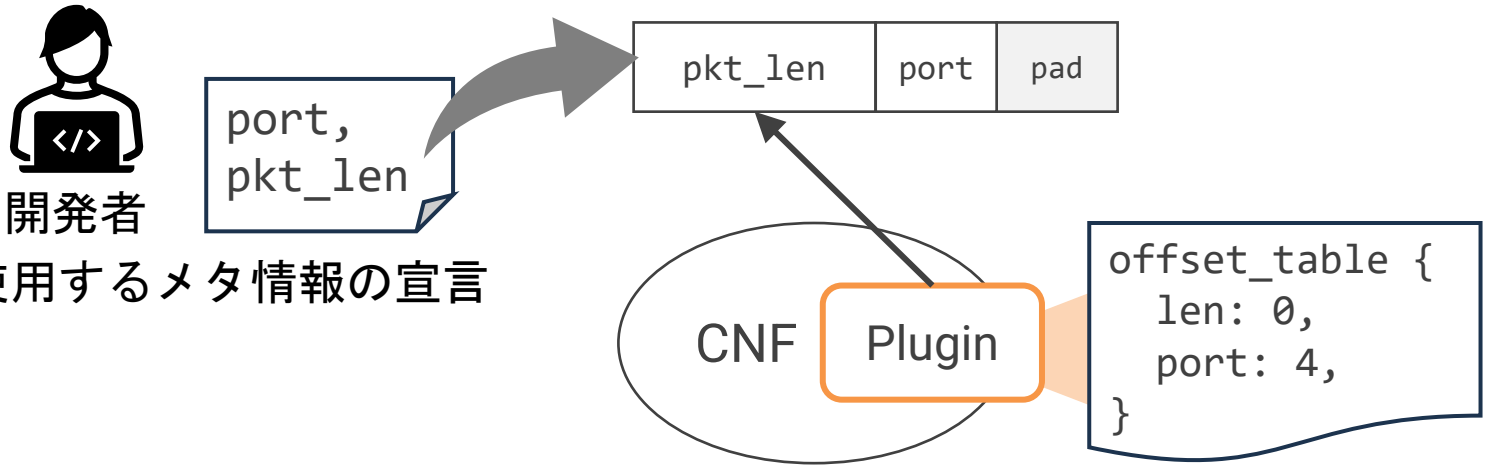
メタ情報領域

- パケット長
- オフロードフラグ
- プロトコルデータ



NF ごとにメタ情報領域を仮想スイッチのランタイム時に構築

→ 仮想・マルチテナント環境のサポート



†山田 他, ”実CNFにおけるメッセージバッファ構成の縮小~キャッシュ効率化に向けた基礎設計~”, NS 研究会’ 2023年度

メタ情報領域縮小の効果

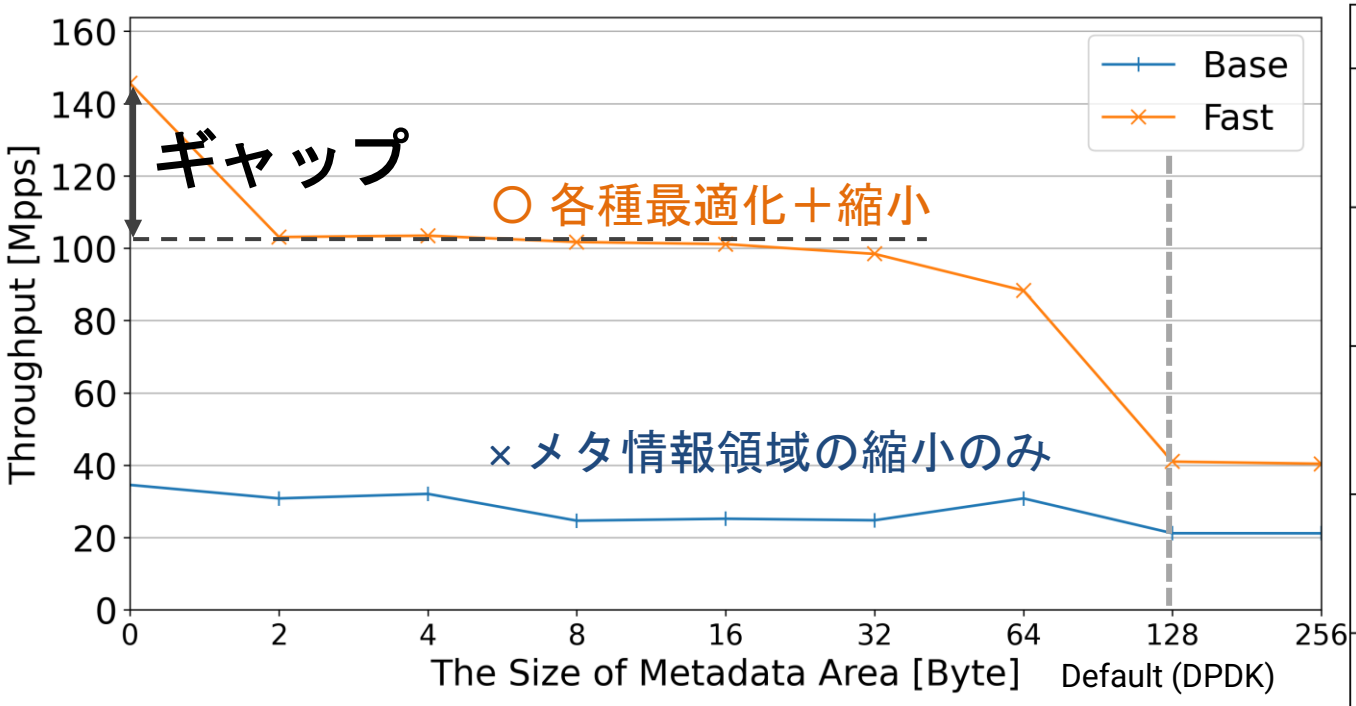
初期化処理の緩和

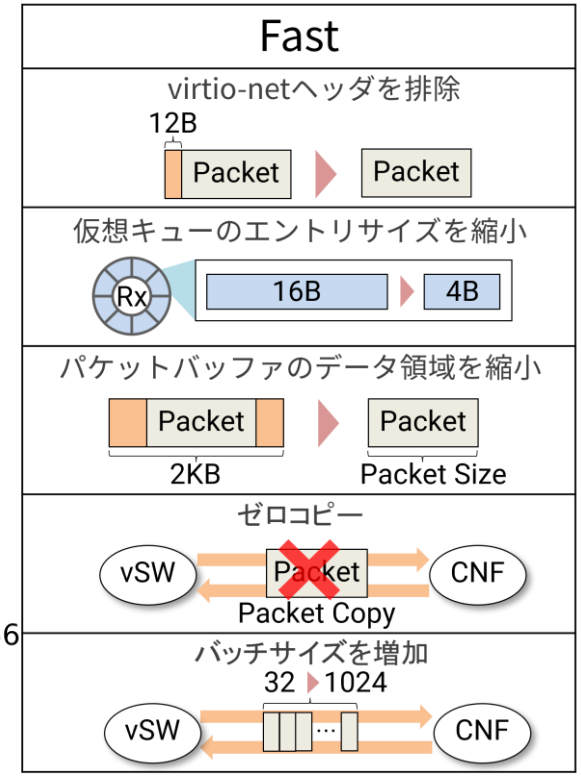
パケット受信時
メタ情報初期化のコスト削減

キャッシュ効率化

キャッシュ消費量の削減

さらなる性能向上のカギ？





ギャップの要因

キャッシュ利用効率

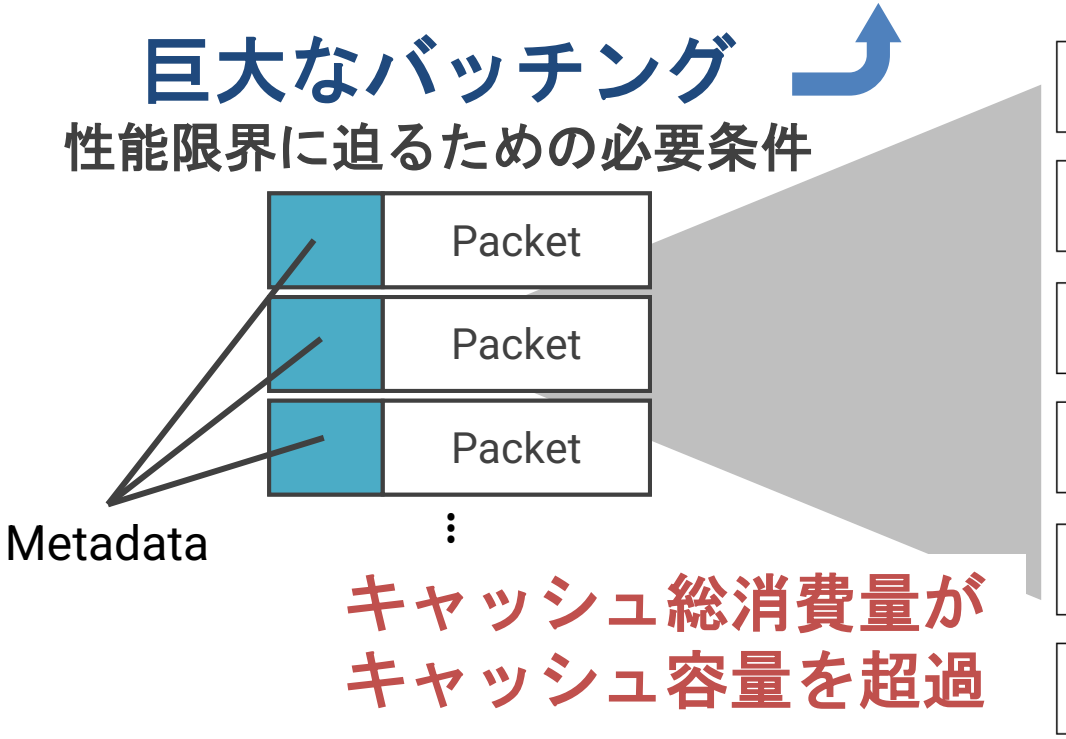
Metadata size	0	2	4	8	16	32	64
Hit ratio [%]	**99.85**	97.73	97.64	97.75	97.68	97.59	97.64
Replace. [times/packet]	**1.79**	5.57	5.64	5.58	5.66	5.74	6.10

先行研究１による理想的な状況

キャッシュライン置換によるヒット率低下

巨大なバッチング

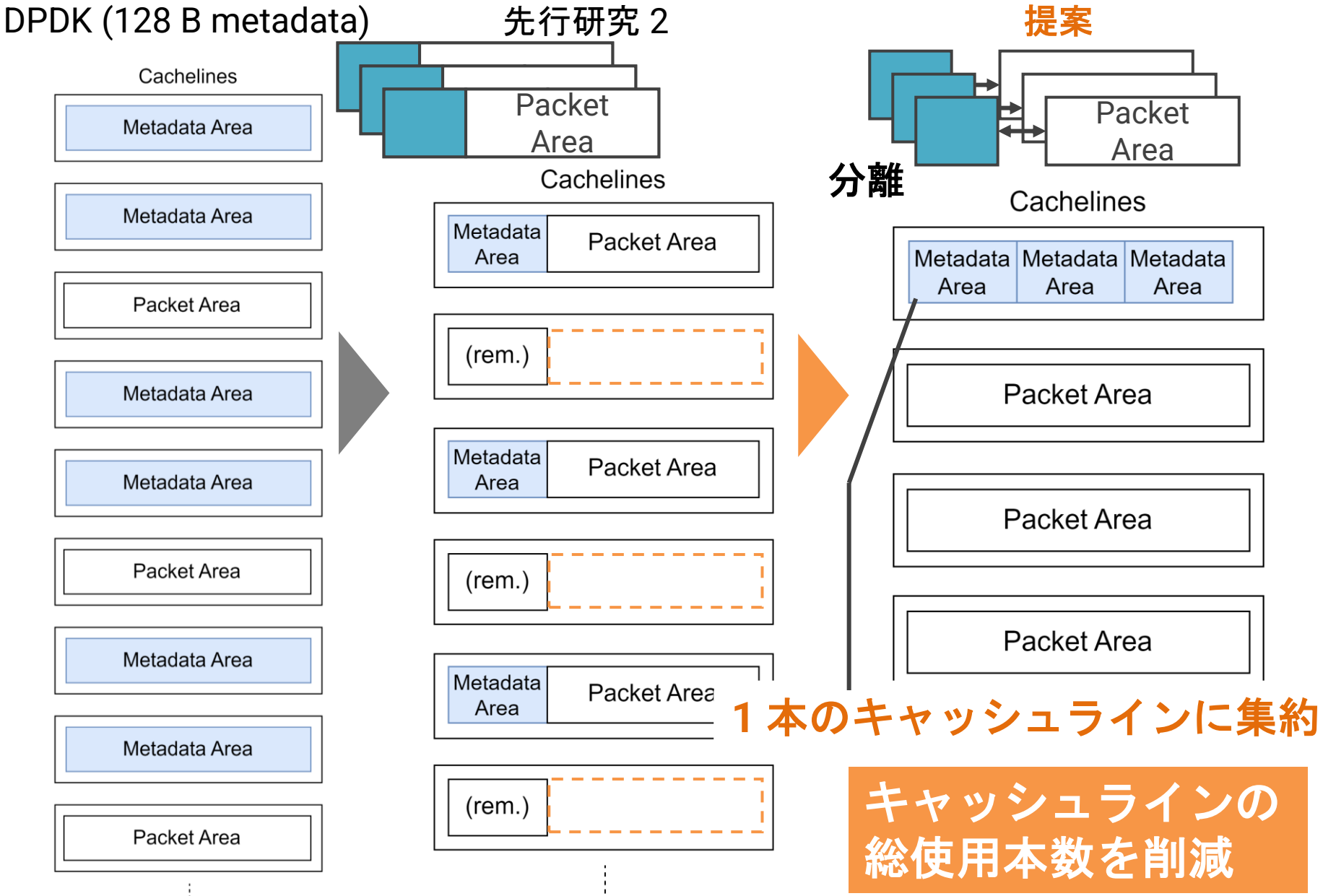
性能限界に迫るための必要条件



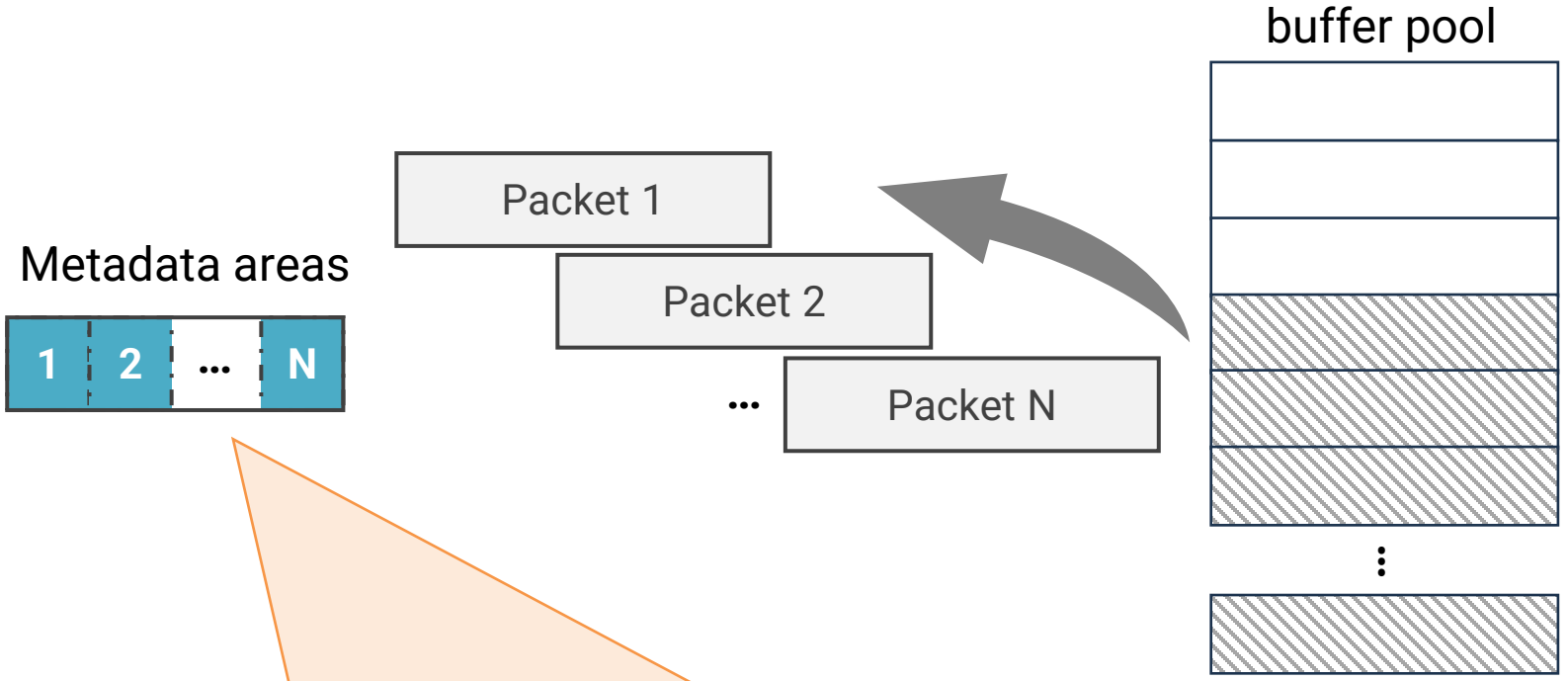
キャッシュ総消費量がキャッシュ容量を超過

空いたスペースを活用すべき

提案手法

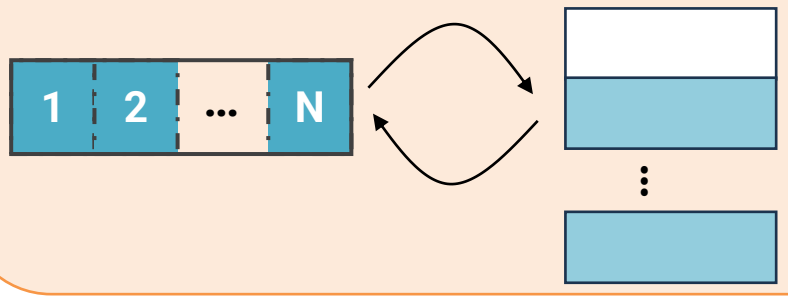

DPDK (128 B metadata)
Cachelines
Metadata Area
Metadata Area
Packet Area
Metadata Area
Metadata Area
Packet Area
Metadata Area
Metadata Area
Packet Area
先行研究 2
Packet Area
Cachelines
Metadata Area
Packet Area
(rem.)
Metadata Area
Packet Area
(rem.)
Metadata Area
Packet Area
(rem.)
提案
Packet Area
分離
Cachelines
Metadata Area
Metadata Area
Metadata Area
Packet Area
Packet Area
Packet Area
１本のキャッシュラインに集約
キャッシュラインの
総使用本数を削減

メタ情報領域の集約

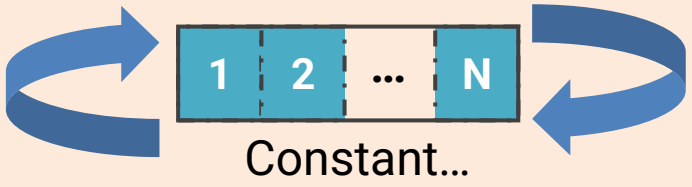


2パターンのアプローチ

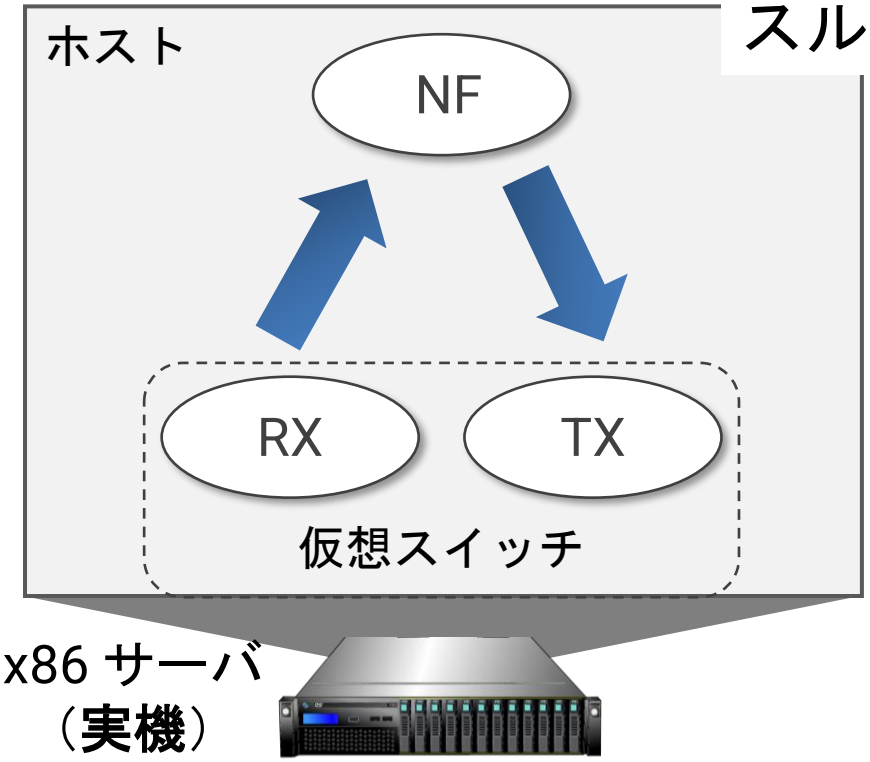
バッファプールから確保・解放 (DPDK)



バッチ長と同数の領域を使い回す



評価環境・評価内容



x86 サーバ
（実機）

CPU	Core i9 11900K
Clock	3.5 GHz
L1d	48 KiB
L2	0.5 MiB
L3 (shared)	16 MiB
Memory Clock	3200 MHz

スループット +

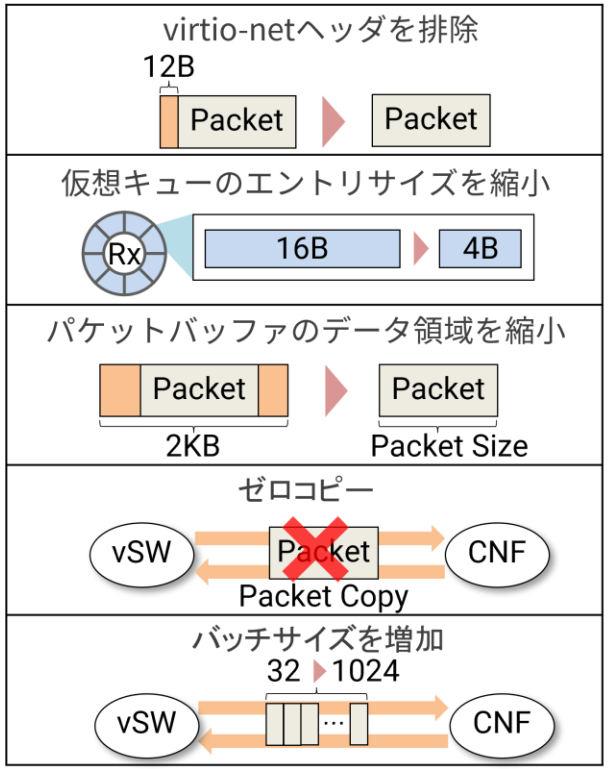
キャッシュ利用状況

- 各キャッシュの総ヒット回数
- 各キャッシュの総ミス数
- キャッシュライン置換回数

…

DPDKの実装・パラメータ

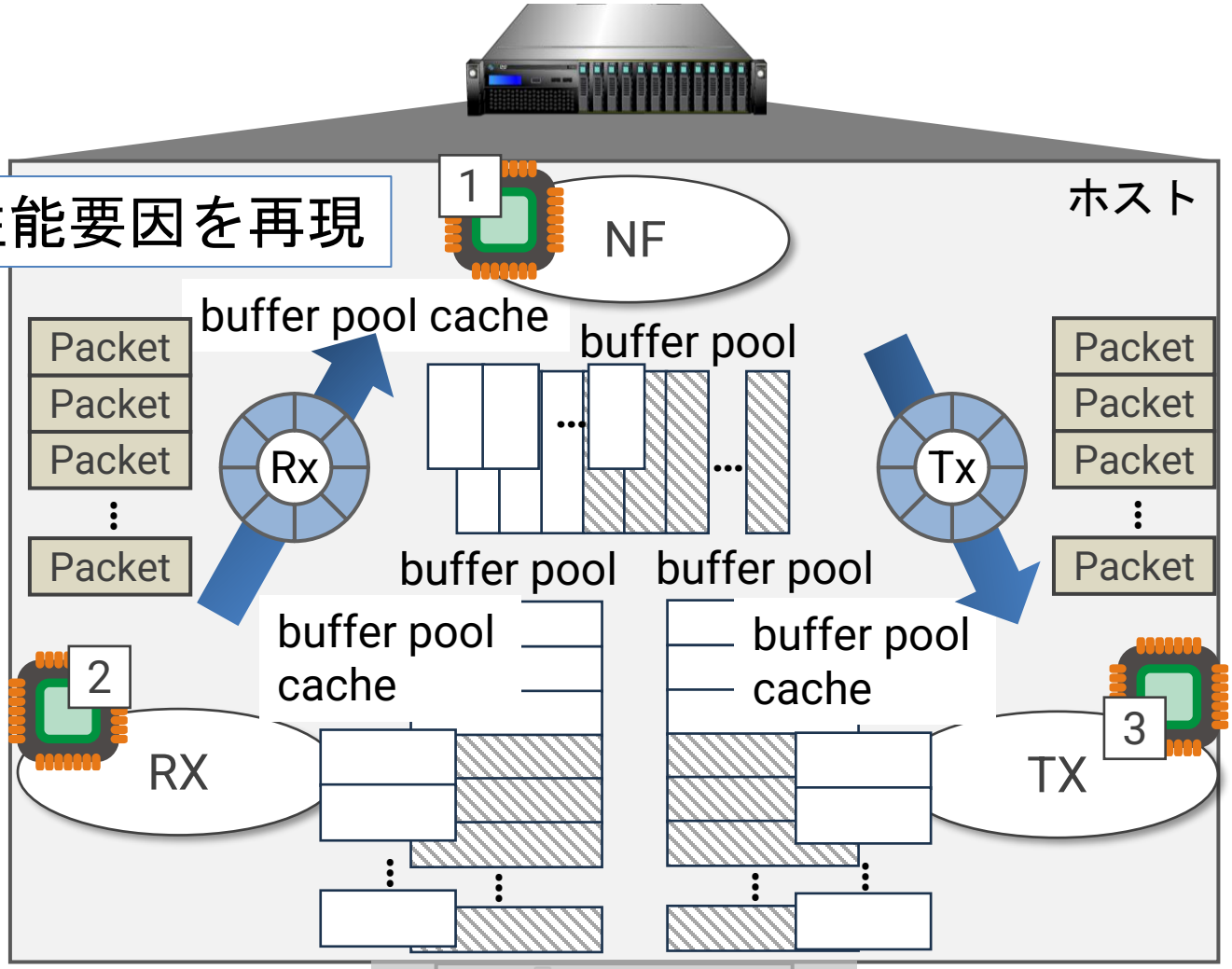
“Base” “Fast”



評価用フレームワーク

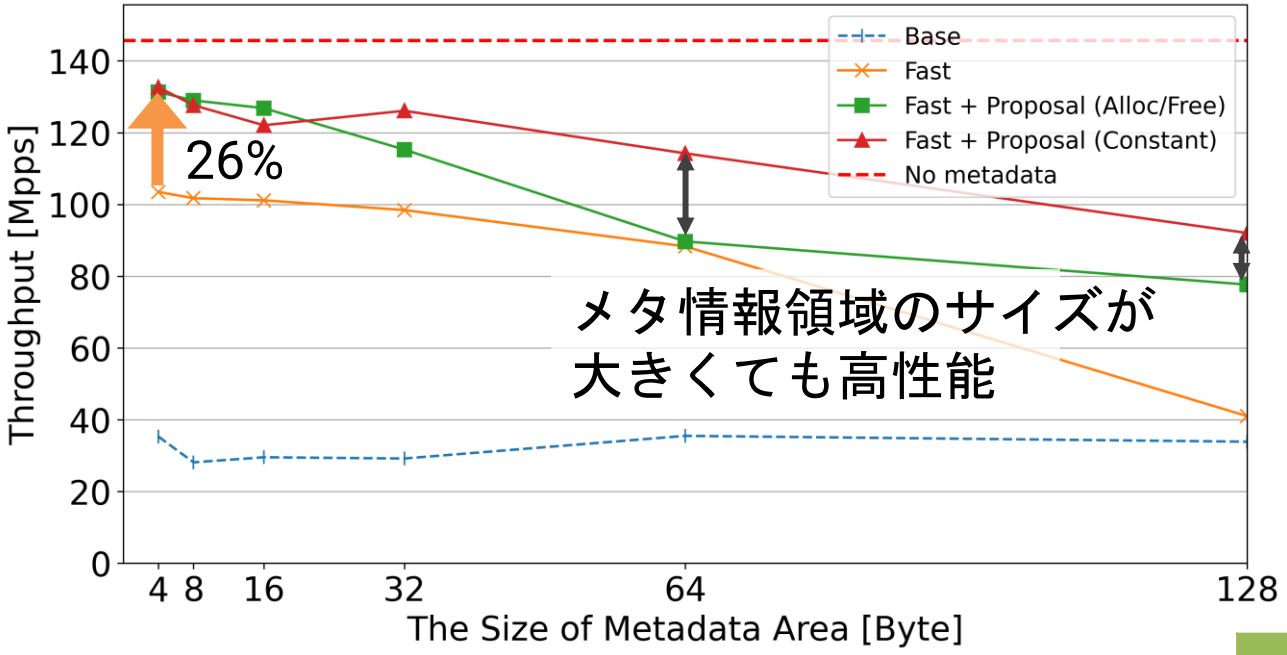
つくりがメッセージバッファ構成と独立→異なる構成を試せる

DPDKの性能要因を再現



NICの影響を受けない

提案手法の効果



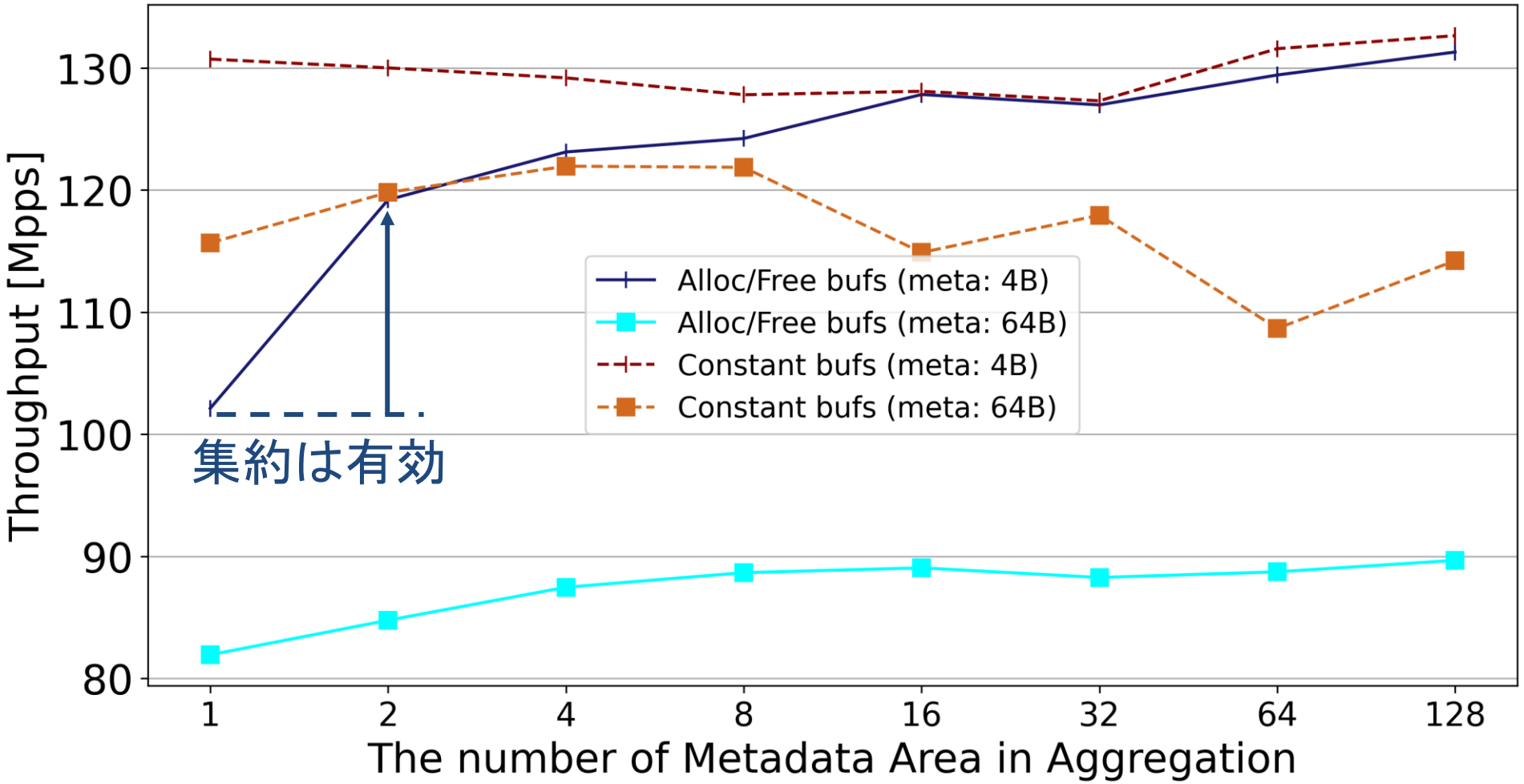
Metadata size	0
Hit ratio [%]	99.85
Replace.[times/packet]	1.79

Fast + Proposal (Alloc/Free)			
4	8	16	64
99.84	99.84	99.85	**98.56**
1.80	1.83	1.83	**3.72**

Fast + Proposal (Constant)			
4	8	16	64
99.89	99.88	99.84	99.83
1.79	1.78	1.82	2.04

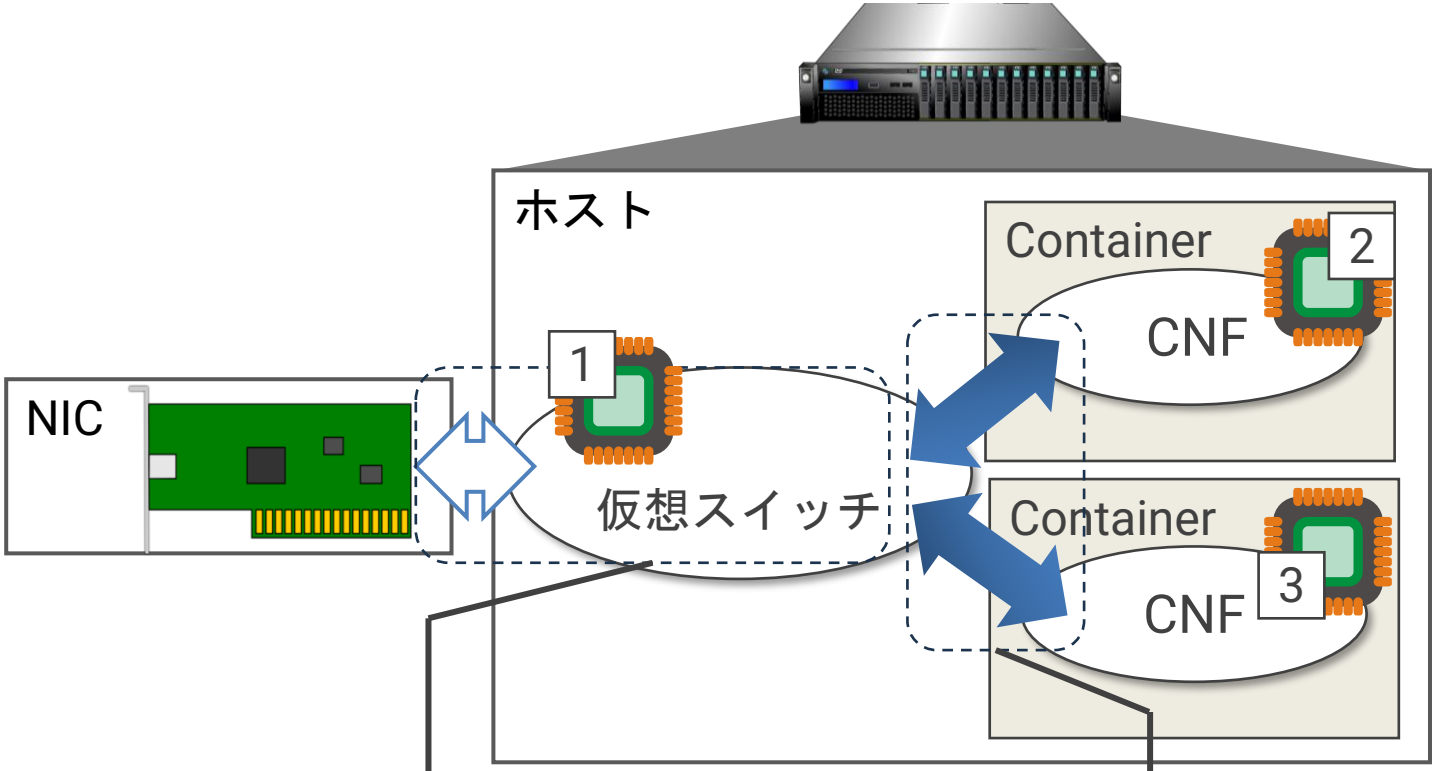
キャッシュ利用効率

メタ情報領域の集約数と性能の関係



集約数が大きいほどスループットが高くなる訳ではない

関連研究



キャッシュ効率化

- NW I/O とNFでLLCを分離[18]
- I/Oキューと連携した
  バッファ管理の簡素化[20]

パケットコピーの回避

- ゼロコピー[5]-[7]
- パススルー[22]

L1キャッシュ利用の効率化に貢献

**単体では
効果はわずか (~ 30 Mpps)**

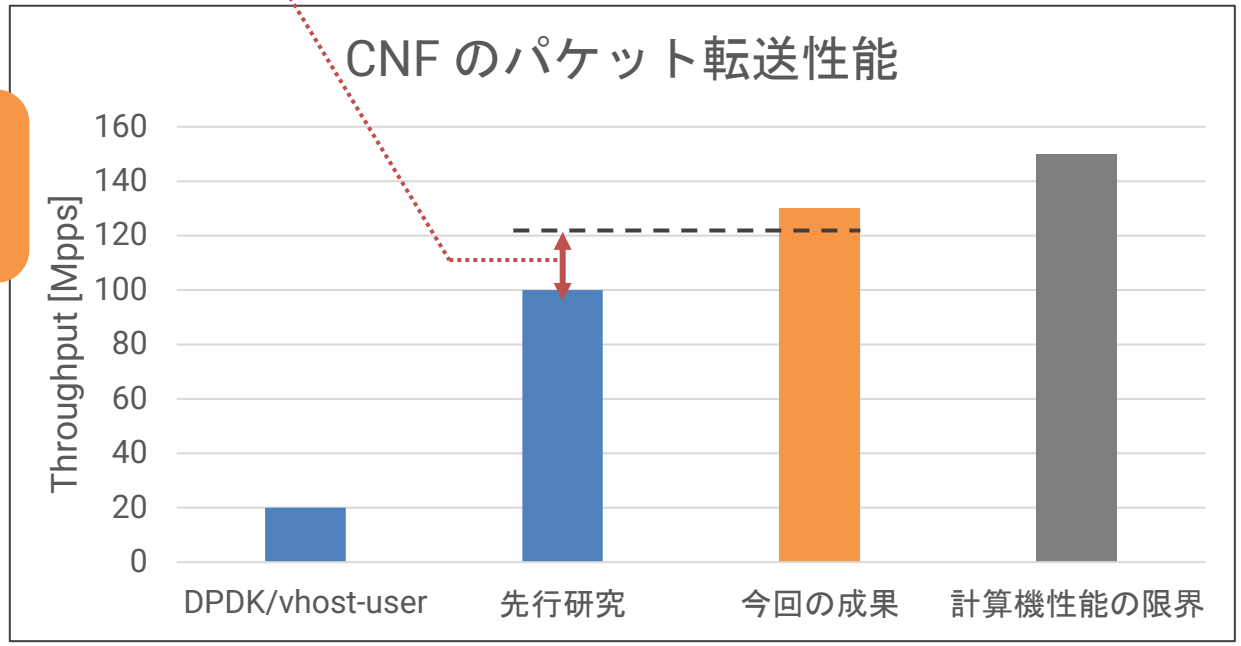
**仮想 NW I/O と
L1キャッシュ利用は考慮外**

まとめと今後の取組

まとめ

巨大なバッチングの際,
キャッシュで 総消費量 > 容量により置換発生 → キャッシュミス

省メモリな
メッセージバッファ構成



今後の取組

- メタ情報領域へのアクセスを一括で行う方式の検討
- 大規模なワークセットを扱うCNFの評価