NTT

lagopus

Innovative R&D by NTT

汎用x86サーバを用いた高速なソフトウエアパケット処理技術

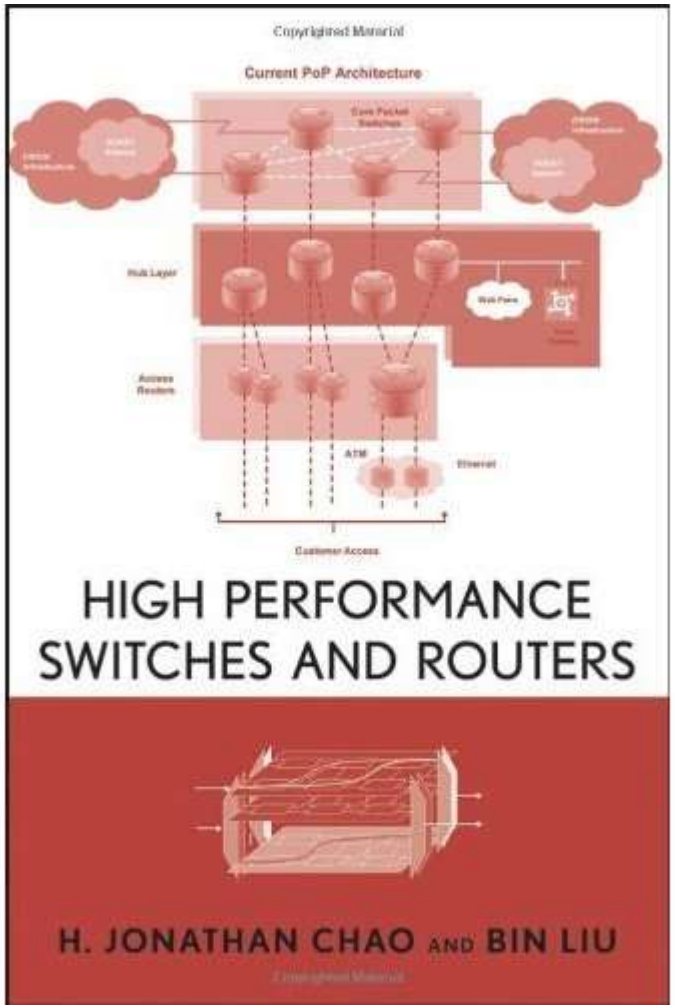
中島佳宏

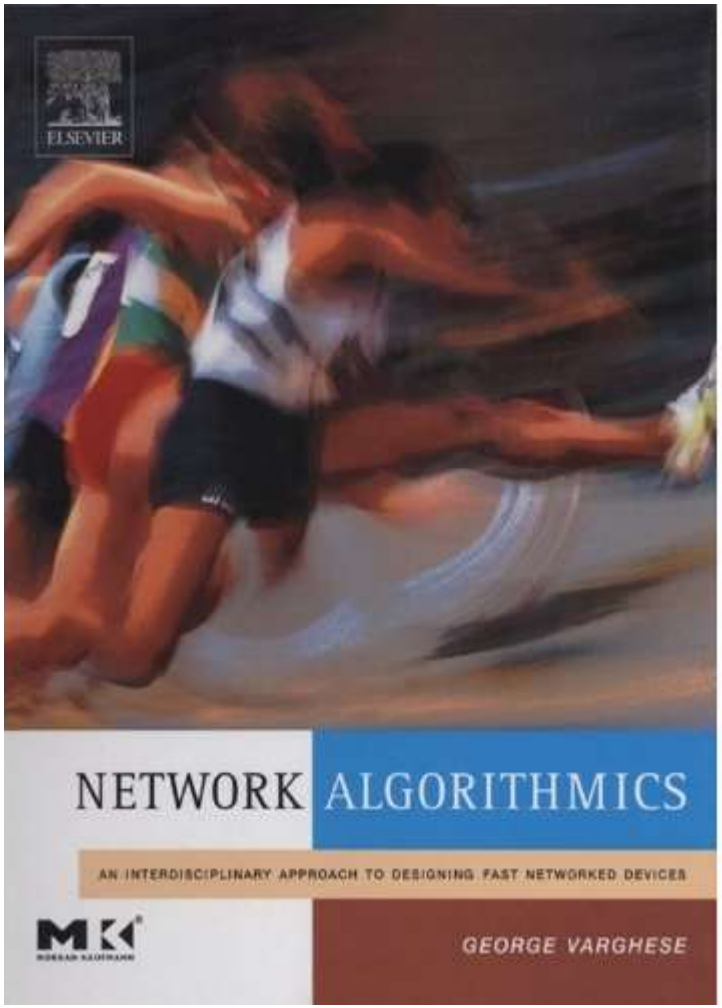
This research is a part of the project for “Research and Development of Network Virtualization Technology” supported by the Ministry of Internal Affairs and Communications.

発表の概要

- パケット処理の流れ
- 汎用x86サーバにおける高速化の難しさ
- 高速化技術
- Lagopus

Two Bible





なぜ汎用x86サーバなのか？

- **汎用サーバはかなり速くなっている**
  - ほぼ毎年性能が改善される
  - コア数もどんどん増えている
- **どこでも入手可能かつ実行可能**
  - 秋葉原や近くの電気屋で買える
  - 仮想環境上でも動作可能
- **安い**
  - 10万以下で8 core CPU, 6 port 1GbEの構成
  - 予算にあわせて様々な構成で買える
- **エンジニアはいっぱいいる**
  - 開発環境・コンパイラ・ライブラリは充実

NTT

lagopus

パケット処理の流れ

packet

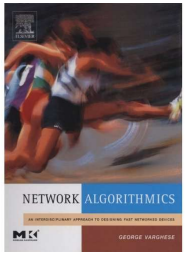
受信処理

NICのドライバ処理

フレーム処理

パケットのパージング

アルゴリズムの
詳細はBibleにて



フロー検索
・ヘッダ書換え

Lookup,
Header書き換え

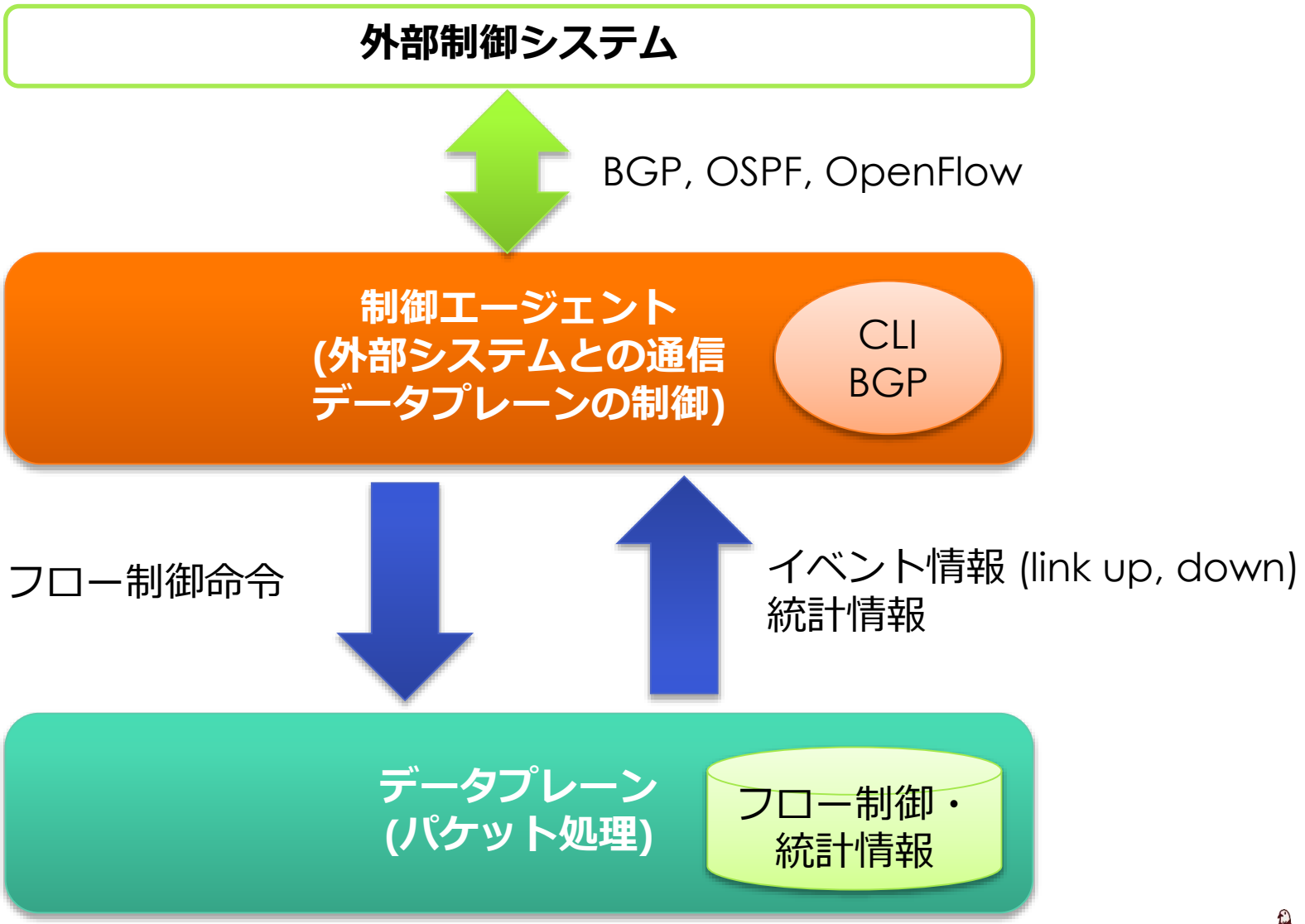
QoS・Queue

Policer, Shaper
Marking

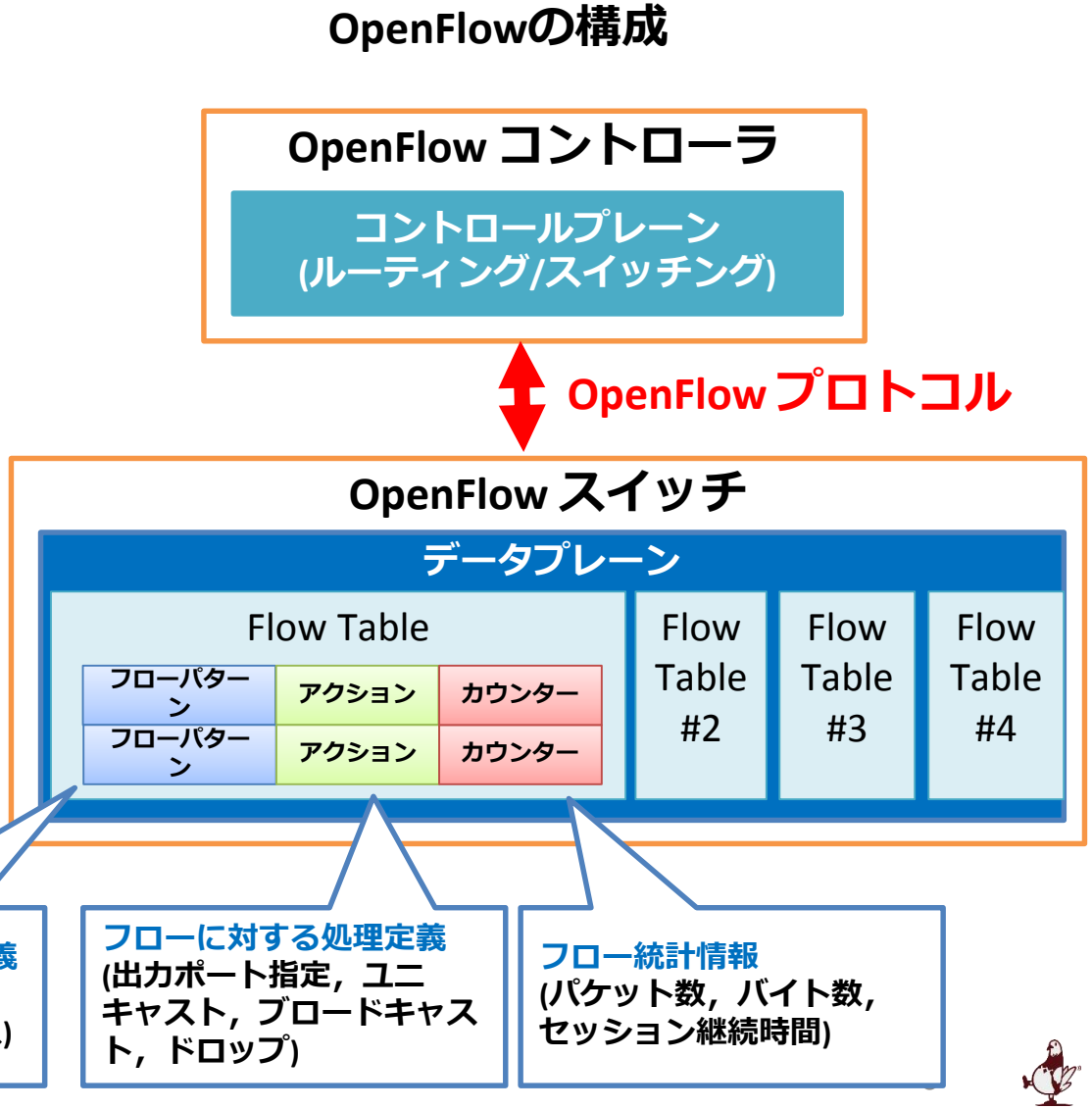
送信処理

packet

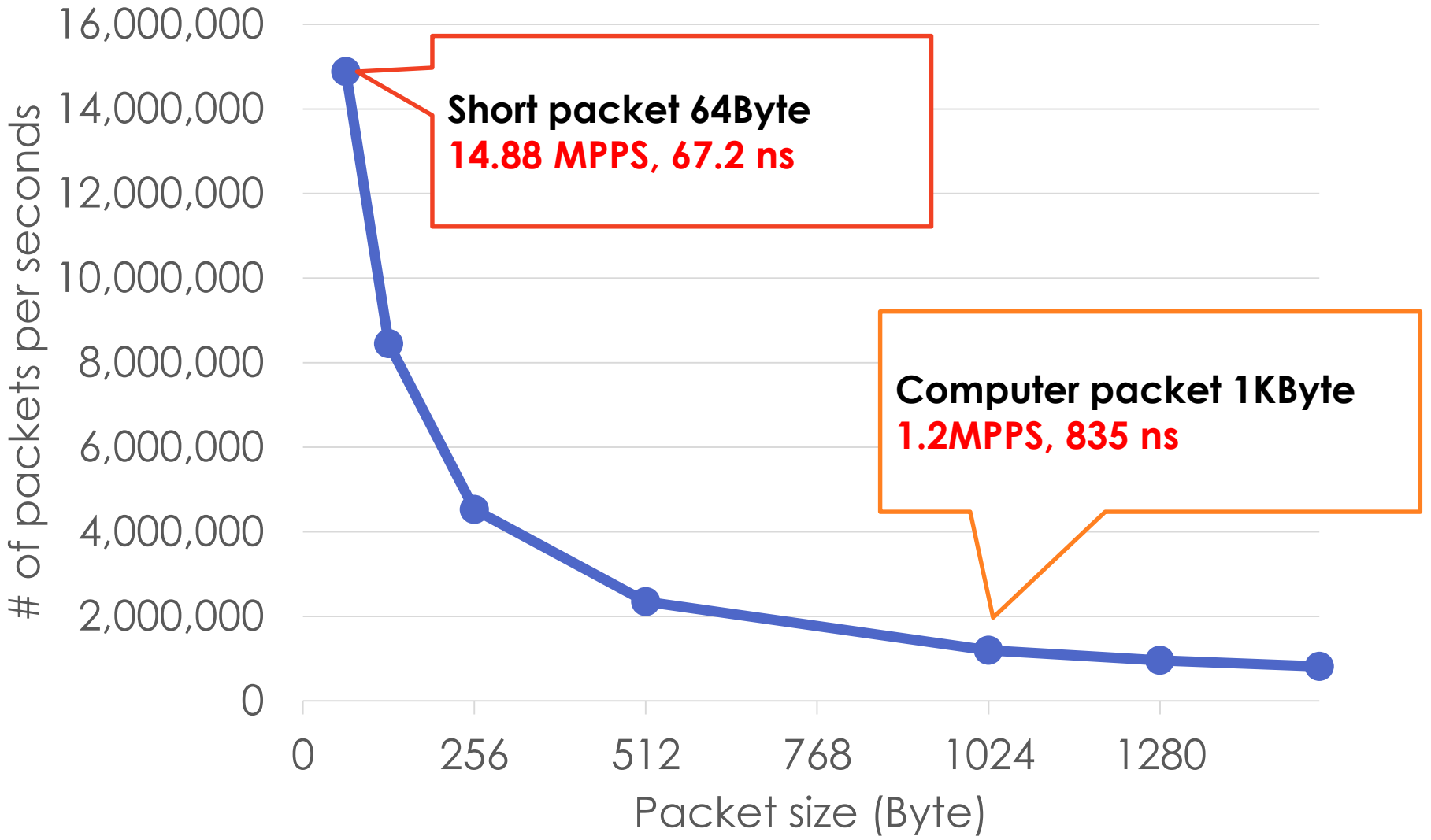
Switchの基本アーキテクチャ



OpenFlowとは (OpenFlow 1.3)

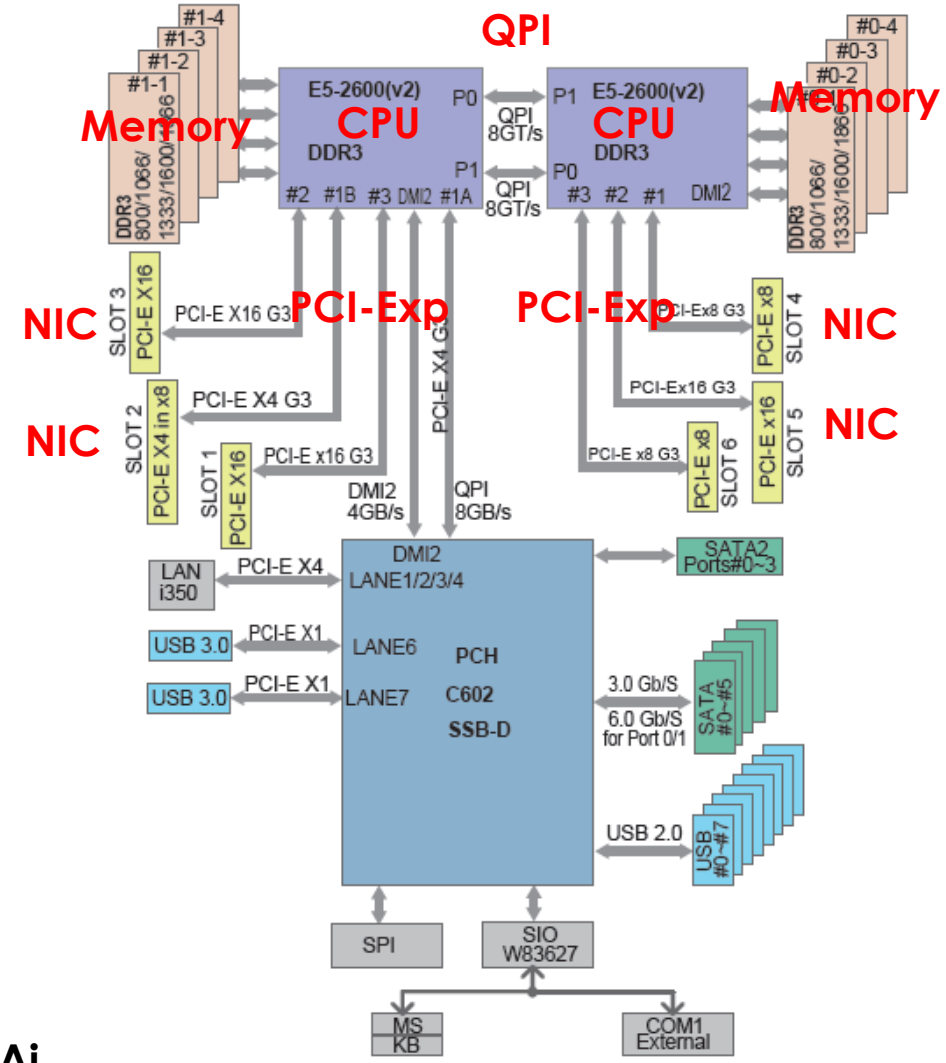


10Gbps達成のために必要なパケット処理数



汎用サーバにおける高速化の難しさ

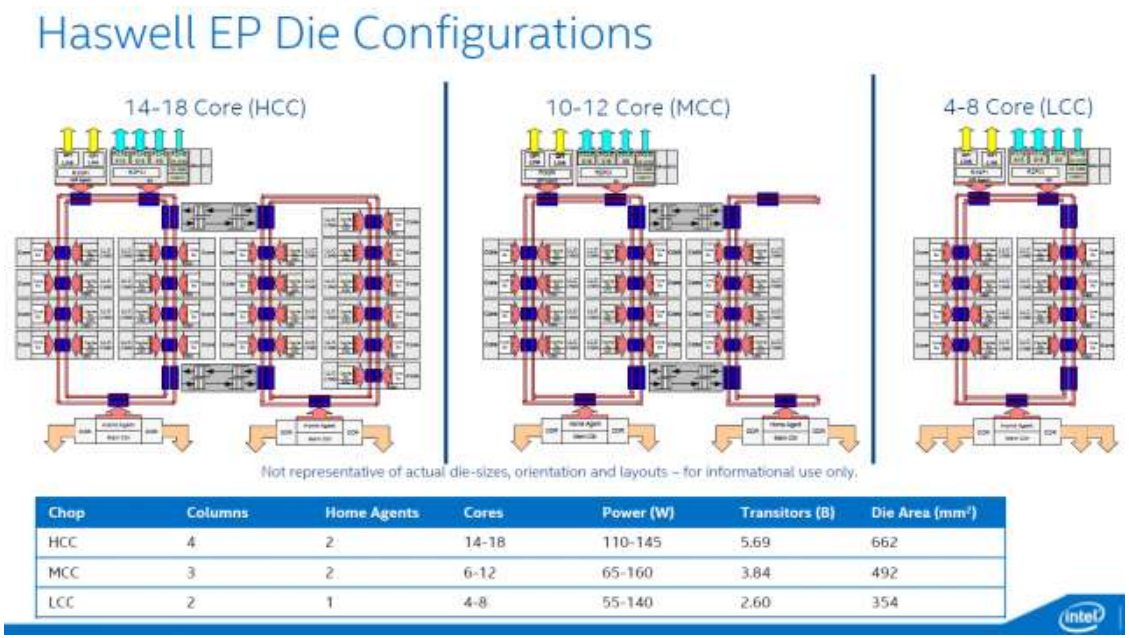
汎用サーバの構成



Reference: supermicro X9DAi

汎用CPUの構成

- **CPUスロットあたり複数CPUコアを保持**
  - 6 – 18CPUコア
- **階層化キャッシュを保持**
  - L1, L2はCPUコア毎に，L3はCPUコア間で共有
- **CPU間はリングで接続**



Chop	Columns	Home Agents	Cores	Power (W)	Transitors (B)	Die Area (mm²)
HCC	4	2	14-18	110-145	5.69	662
MCC	3	2	6-12	65-160	3.84	492
LCC	2	1	4-8	55-140	2.60	354

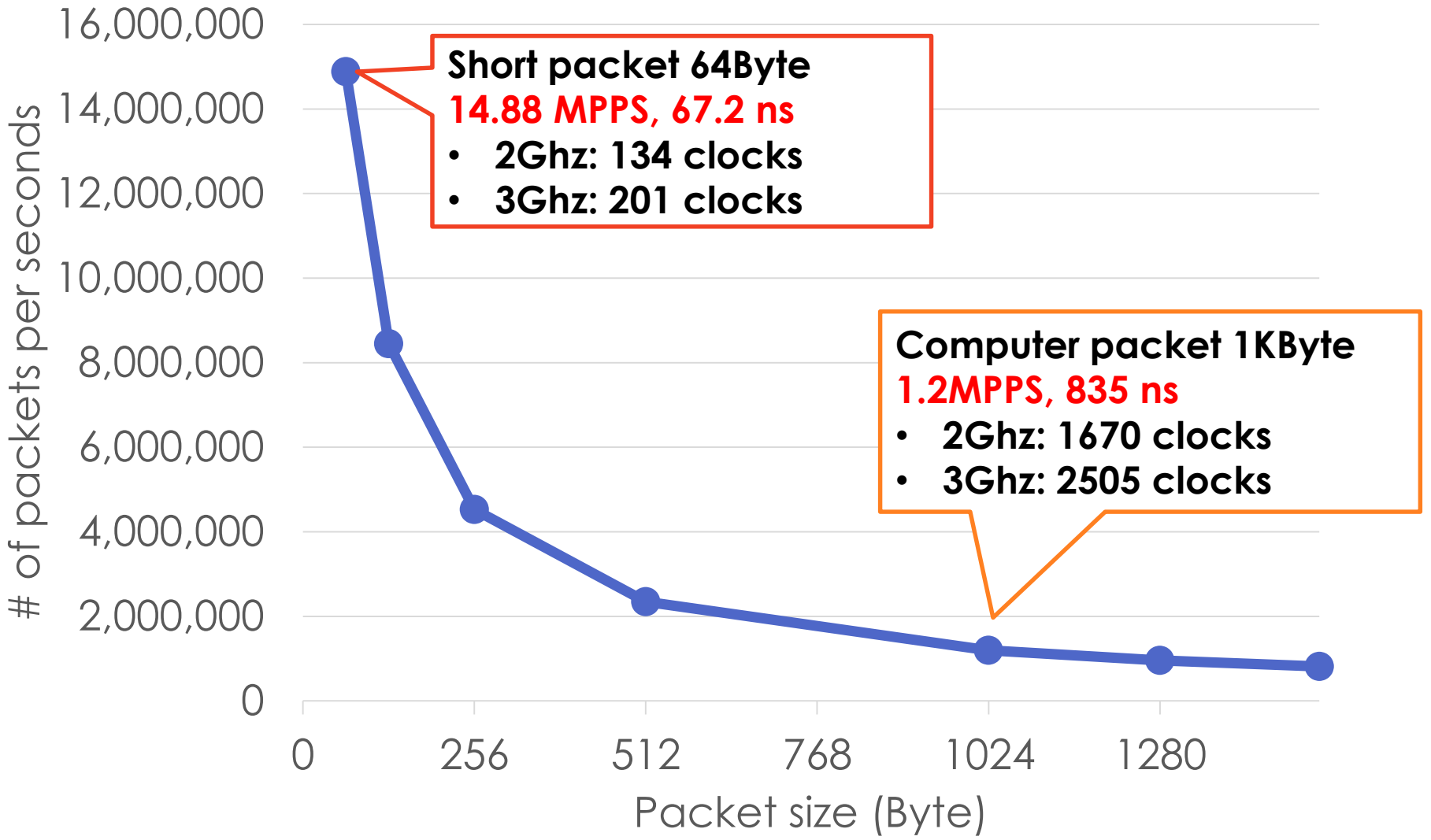
intelさん資料より

CPUのキャッシュとメモリの構成

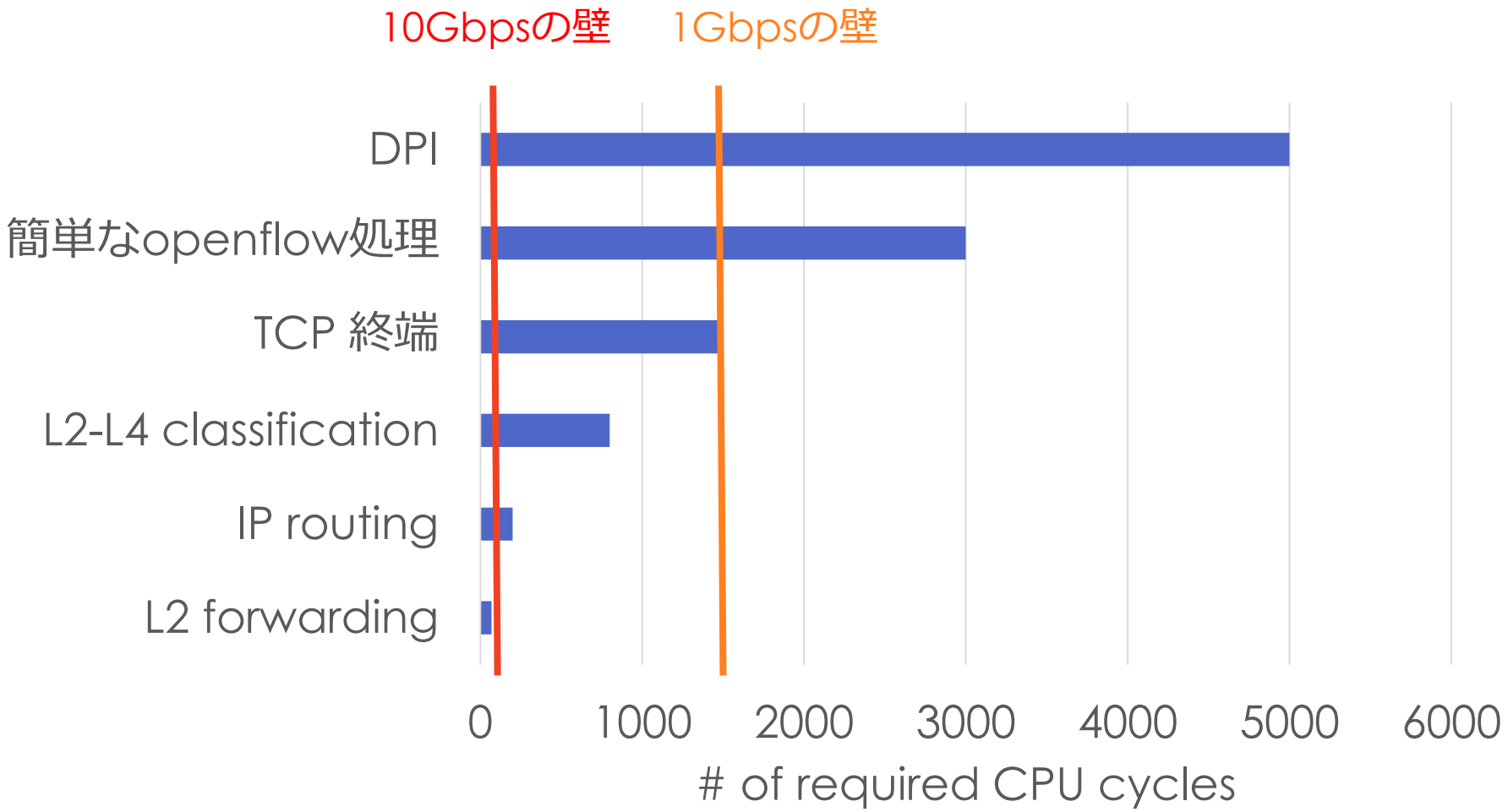
- **階層化されたキャッシュを持っている**
  - キャッシュに入っているデータ・アクセスはとても速い
  - メインメモリのデータを見る場合はとても遅い
- **連続するデータについてはプリフェッチ機構が利用可能**

	Size (Byte)	Bandwidth (GB/s)	Latency - random
L1 $	32K	699	4 clock
L2 $	256K	228	12 clock
L3 $	6M	112	44 clock
Memory	G class	20	300 cycles (91 ns)

10Gbps達成のために必要なパケット処理数

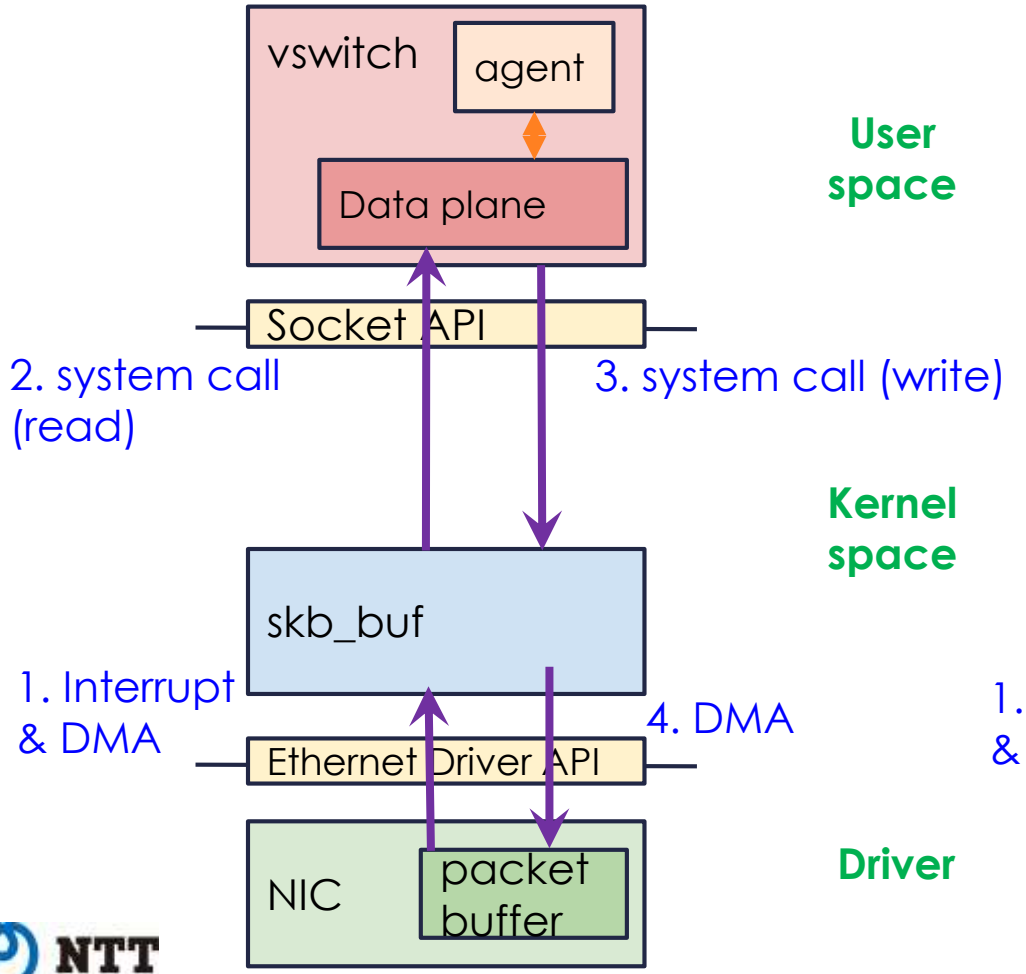


典型的なパケット処理にかかるCPUサイクル数 (2Ghz CPU)



パケット処理アプリ on Linux/PC

ユーザ空間でのパケット処理アプリ (イベントベースの処理)

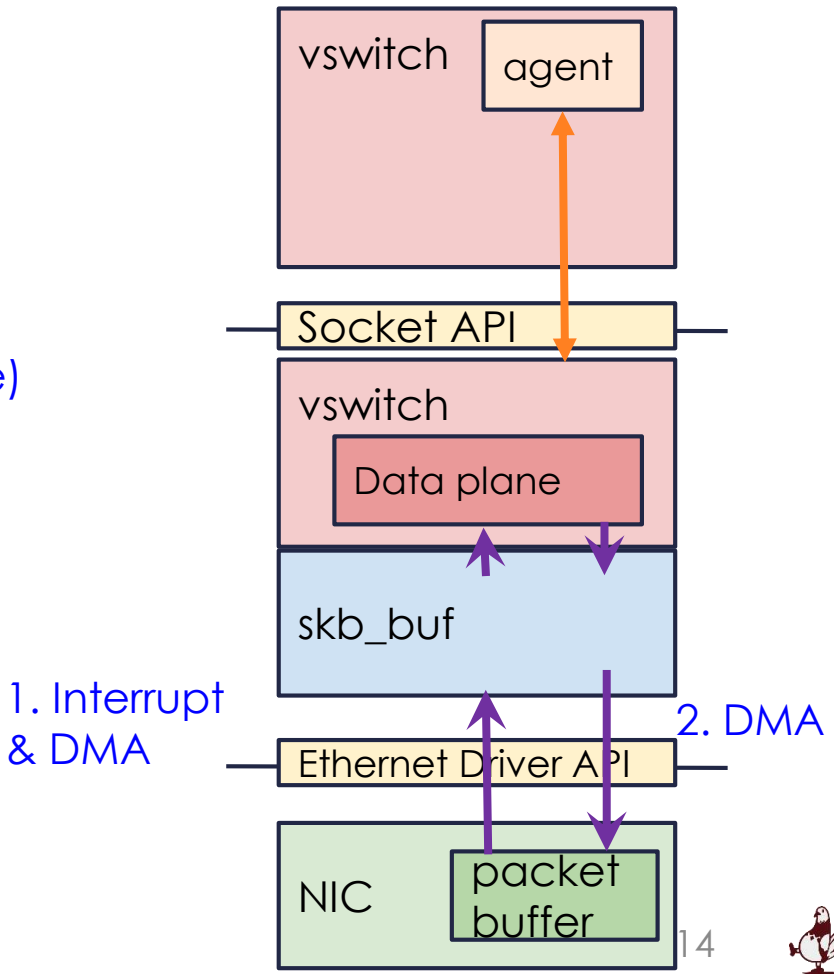


User space

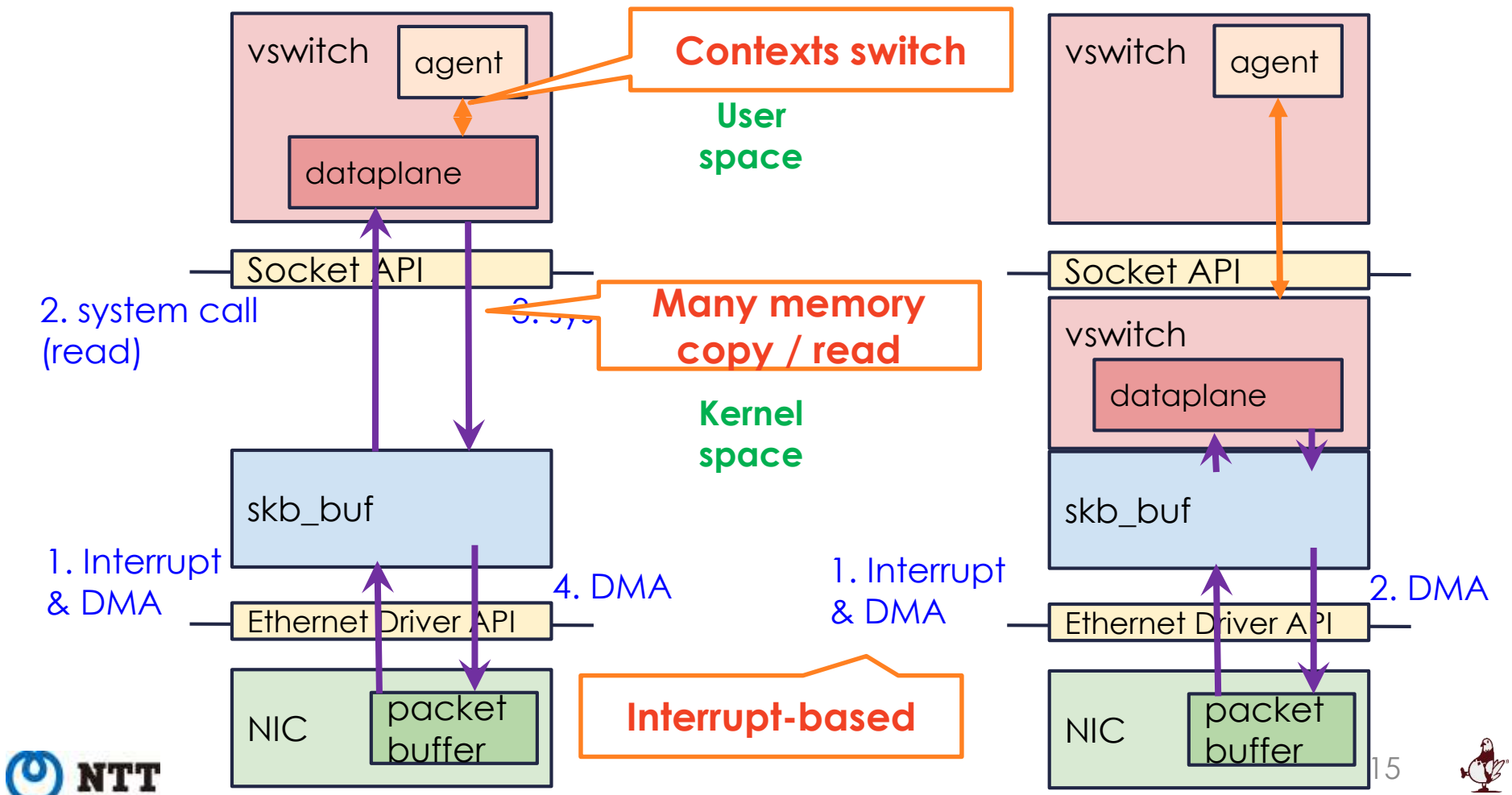
Kernel space

Driver

カーネル空間でのパケット処理 (イベントベースの処理)



パケット処理アプリ on Linux/PC



汎用x86での高性能パケット処理の課題

1. 高負荷時の大量の受信割り込み処理にサーバは耐えられない
2. 他のプロセスの介入などのタスクスイッチのオーバヘッドが重い
3. CPUと比較してPCI-Express I/O, memoryのバンド幅が狭い
4. 複数スレッド時に共有データへの排他処理がボトルネック
5. メモリアクセス単位が4KBのため，大量メモリアクセスに伴うアドレス変換機構 (TLB) のキャッシュ頻繁にスワップアウト (非効率)

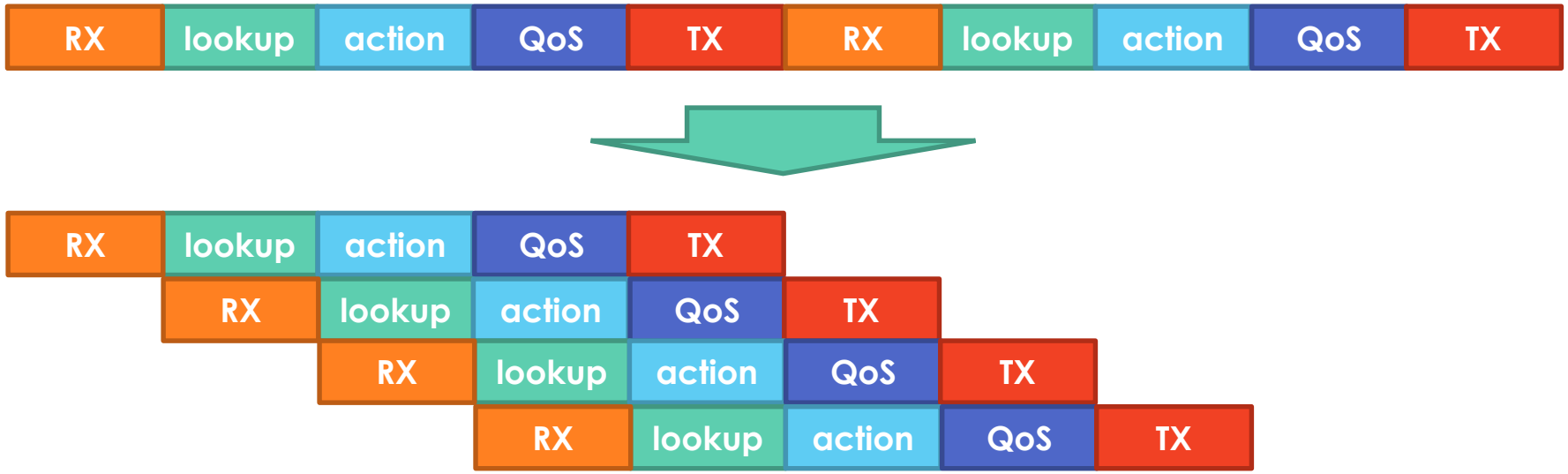
高速化のための技術

高速化のための技術

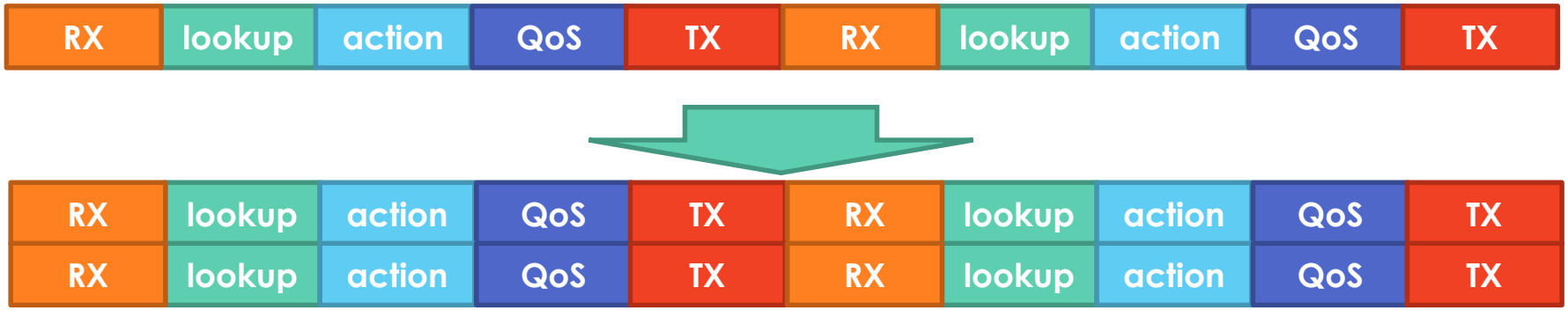
- **マルチコアCPU活用**
  - パイプライン処理
  - 並列処理
  - マルチスレッド化
- **高速なI/Oライブラリ**
  - Intel DPDK
  - Netmap
- **ロックレスライブラリ**
  - RCU
- **高速なフロー検索**
  - 詳しくはBibleを参照

高速化のための基本的なアイディア (1)

■ パイプライン化



■ 並列化

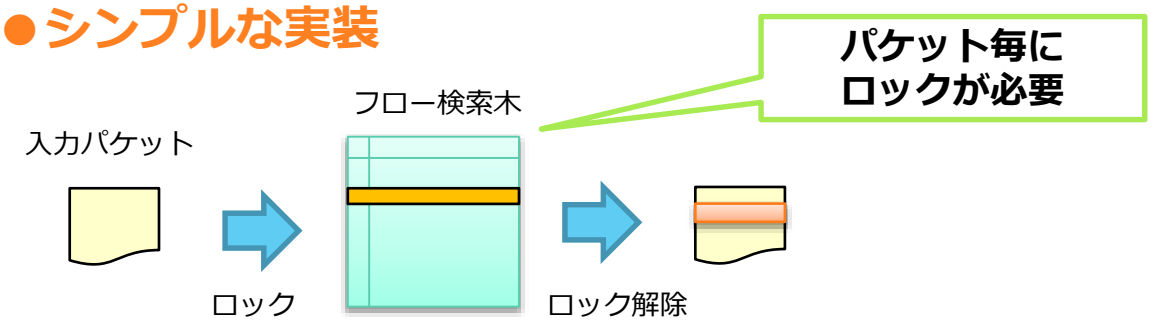


高速化のための基本的なアイディア (2)

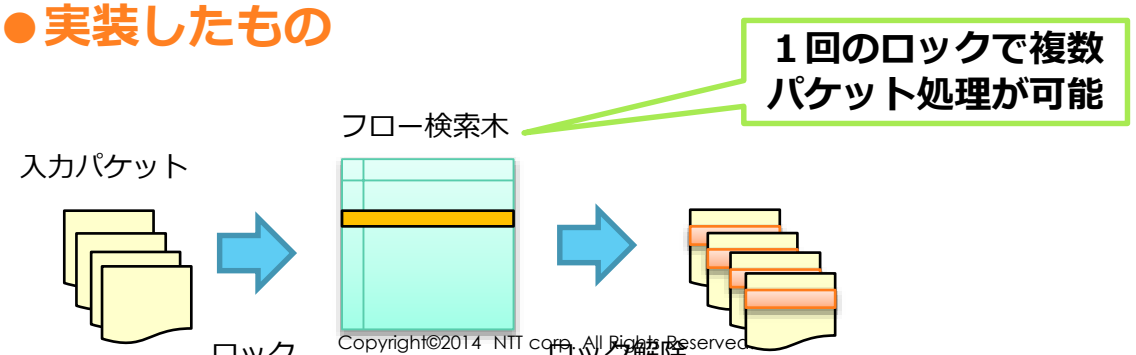
■ パケットバッチング

- パケット毎にフロー検索と処理を行うのではなく，パケットを複数まとめて実行
  - CPUのメモリプリフェッチが活用可能
  - 共有データに対するロック削減

●シンプルな実装



●実装したもの



高速化のための実装の工夫

■ 処理のバイパス

- 受信時パケットデータをNICのバッファから直接CPUキャッシュへ転送 (Intel Data Direct I/O)
- カーネルにおける処理のバイパス
- フロー検索$活用によるフロー検索処理の軽量化

■ メモリアクセスの削減と局所化

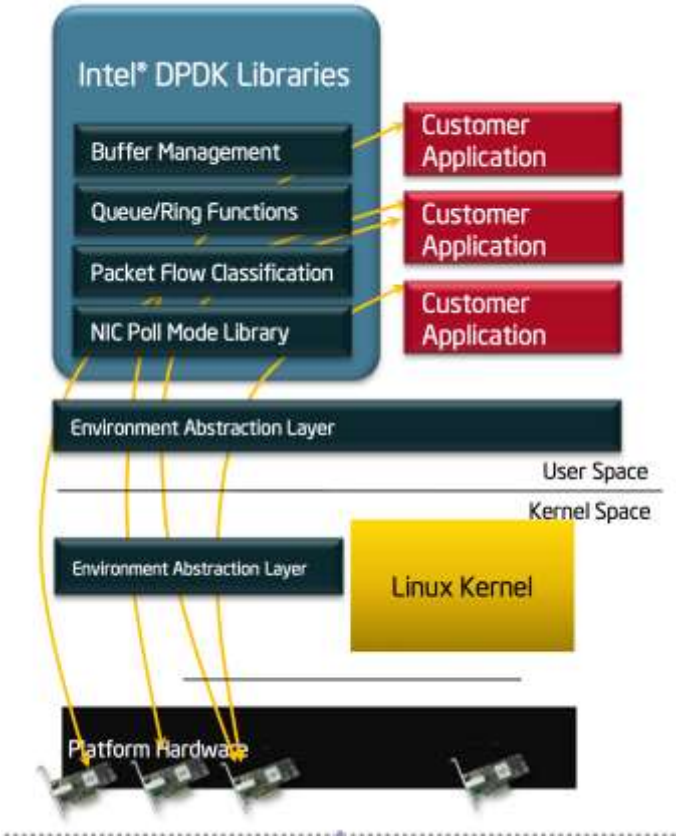
- コンパクトな検索木の作成
- Thread local storageの活用
- パケット処理の部分化とそのCPUへの明示的な割当て

■ パケット処理のCPUコアへの明示的な割り当て

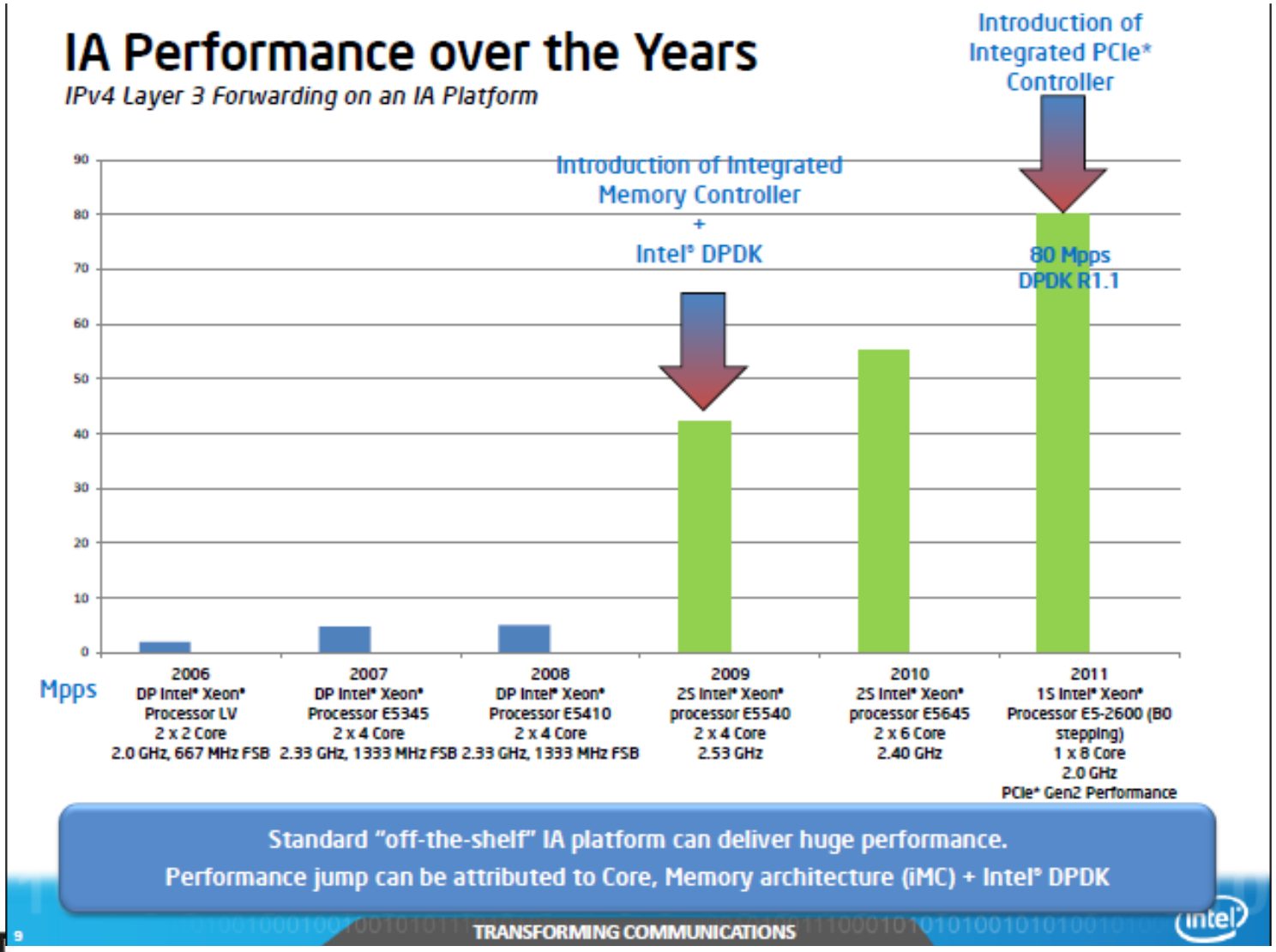
- CPUやバスの内部接続を考慮したスレッドのCPUへの割り当て

Intel DPDK

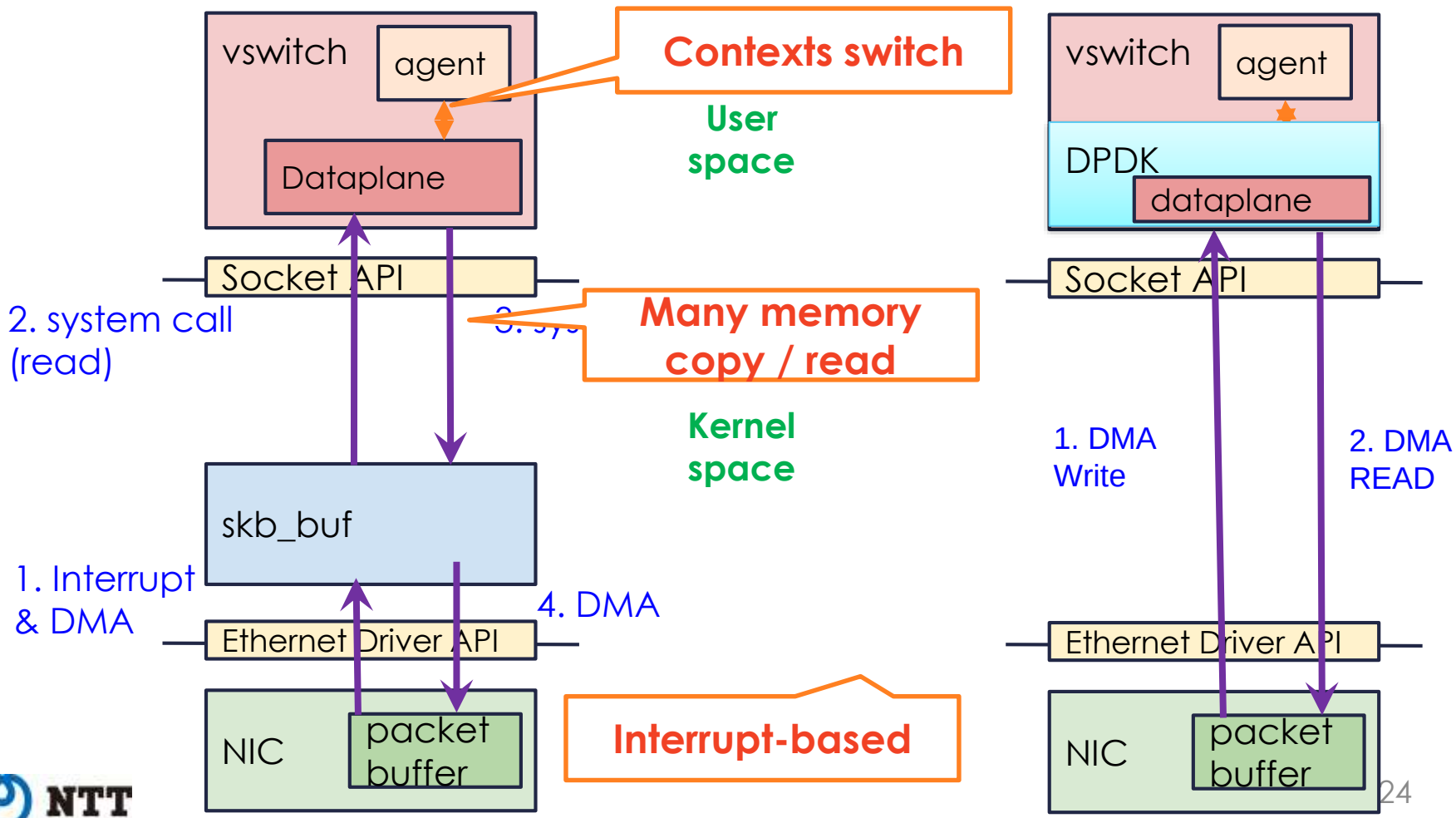
- **X86アーキテクチャに最適化されたデータプレーン用ライブラリとNICドライバ**
  - メモリ構成を考慮したQueue, buffer管理ライブラリ
  - packet flow classification
  - ポーリングベースのNIC driver
- **データプレーン処理に最適化された低遅延 & 高性能なランタイム**
- **抽象化されたパケット処理**
- **BSDライセンス :)**
  - OSSコミュニティにより開発中



Intel DPDKの効果



DPDK パケット処理アプリ



汎用x86上での高性能パケット処理の課題と解決法

1. 高負荷時の大量の受信割り込み処理はサーバにとって辛い
   → ポーリングベースのパケット受信処理
2. 他プロセスによるタスクスイッチのオーバヘッドが重い
   → スレッドの明示的なCPU割り当て
   (one thread/one logical CPU)
3. PCI-Express I/O, memoryのバンド幅が狭い
   → I/Oやメモリへのアクセス回数を可能な限り削減
4. 複数スレッド時に共有データへの排他処理がボトルネック
   → ロックレスキューやRCUライブラリ活用
5. メモリアクセス単位が4KBのため，大量メモリアクセスに伴うアドレス変換機構 (TLB) が非効率
   →Huge DTLB (2MB - 1GB)

NTT

lagopus

Lagopus: 高速なソフトウエア OpenFlow Switch

なぜ始めたのか？

■ 高性能なソフトスイッチが欲しかった

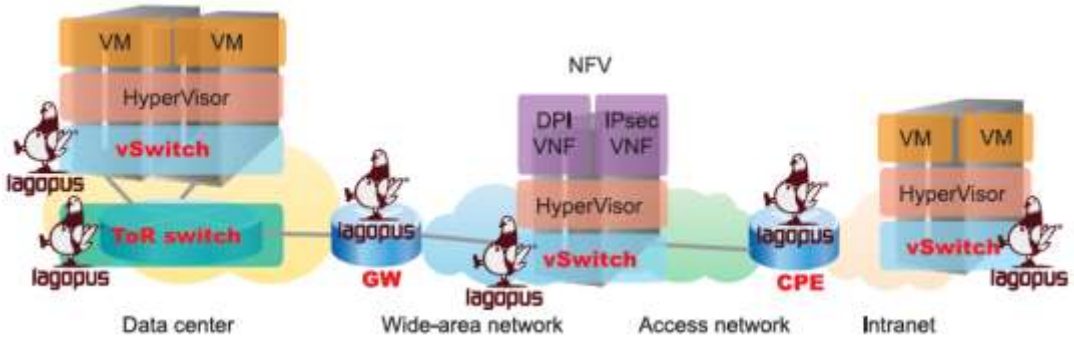
- 機能面
  - MPLSサポートしてほしい ☹
  - PureなOpenFlowスイッチが欲しい
- 性能面
  - 10万フロー投入に数時間かかる ☹
  - 思ったよりパケット転送が遅い☹
- 管理面
  - インストールがしやすいほうがいい
  - ユーザ空間で動作するほうがいい

背景

- **Agile and flexible networking**
  - 自動化の重要性
  - ユーザ手動のNW構成設定
- **サーバ仮想化とNFVからの要求により高性能なソフトウエアによるパケット処理が必要**
  - レイテンシを可能な限り小さく
  - ショートパケット (64B) 時でも高い性能
- **まともなOpenFlow 1.3ソフトスイッチがない**
  - 大規模なフローテーブル
  - 10Gbs回線
  - 容易なアップデートや管理の容易化の促進

Lagopusのターゲット

- **高性能なソフトウエアOpenFlow switch**
  - 10Gbps wire-rate packet processing / port
  - 1M flow rules
  - 高速なフローの投入
- **SDNの広域NWへの適用**
  - data center以外の領域へのチャレンジ
  - WAN protocols, e.g. MPLS and PBB
  - 管理効率化のための管理プロトコル対応
- **ゲートウェイシステムや仮想化基盤向けのvSW**



Lagopus
設計・実装の方針

設計・実装の方針 (1)

■複雑にしたら負け

- ●複雑骨折しない
- ●シンプルな方がCPUの高速化機能を利用できる
- ●OpenFlow 1.0をサポートしない
- ●教科書に従いフルスクラッチ開発

設計・実装の方針 (2)

■メモリコピーしたら負け

- ●一度NICからメモリコピーしたら動かさない
- ●可能な限りメモリコピーしない

■ロックしたら負け

- ●パケットバッチ処理を可能な限り取り入れる
- ●可能な限りロックしないようサボる

■なんでも交換可能

- ●フロー検索，ライブラリとか
- ●研究プラットフォームとして利用可能なように

Lagopus設計

■ シンプルなモジュール構成

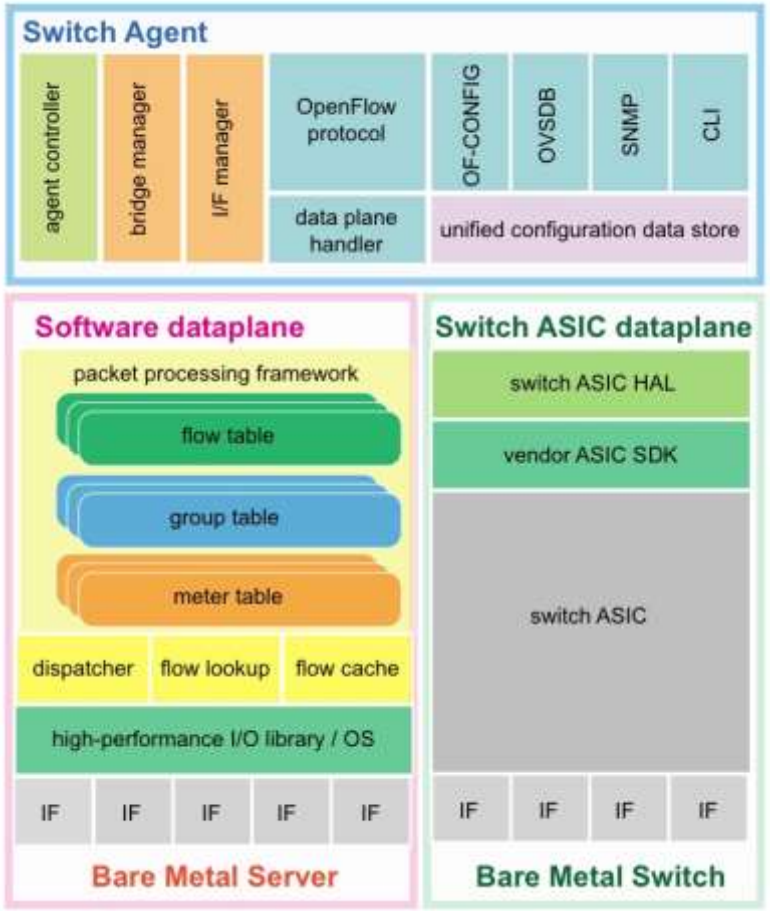
- スイッチ制御
- データプレーン

■ Switchエージェント

- 統一スイッチ資源モデル
- HALを介したデータプレーン制御 (Event Queueベース)

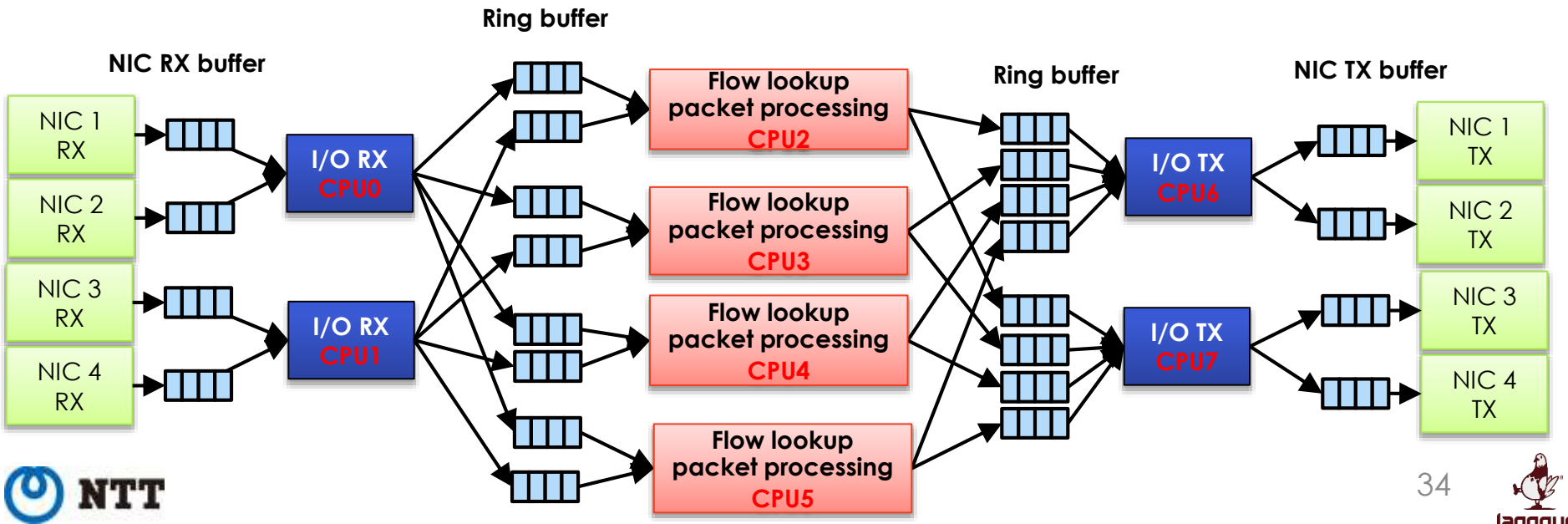
■ データプレーン

- 高速NW I/Oライブラリ (Intel DPDK)
- 複数フローテーブルに対応したフローキャッシュ



複数CPUコアによるパケット処理

- **OpenFlow data plane processing**
  - パケットはコピーせず参照渡し
  - パケット群に対するバッチ処理
- **Exploit many core CPUs**
  - I/O処理とFlow処理を分離
  - マルチコアCPUを利用したパイプライン化
  - CPU D-cache利用効率を向上

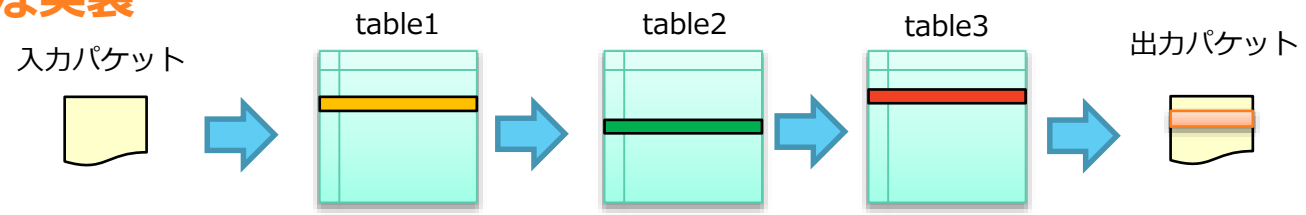


バイパスするためのキャッシュ機構

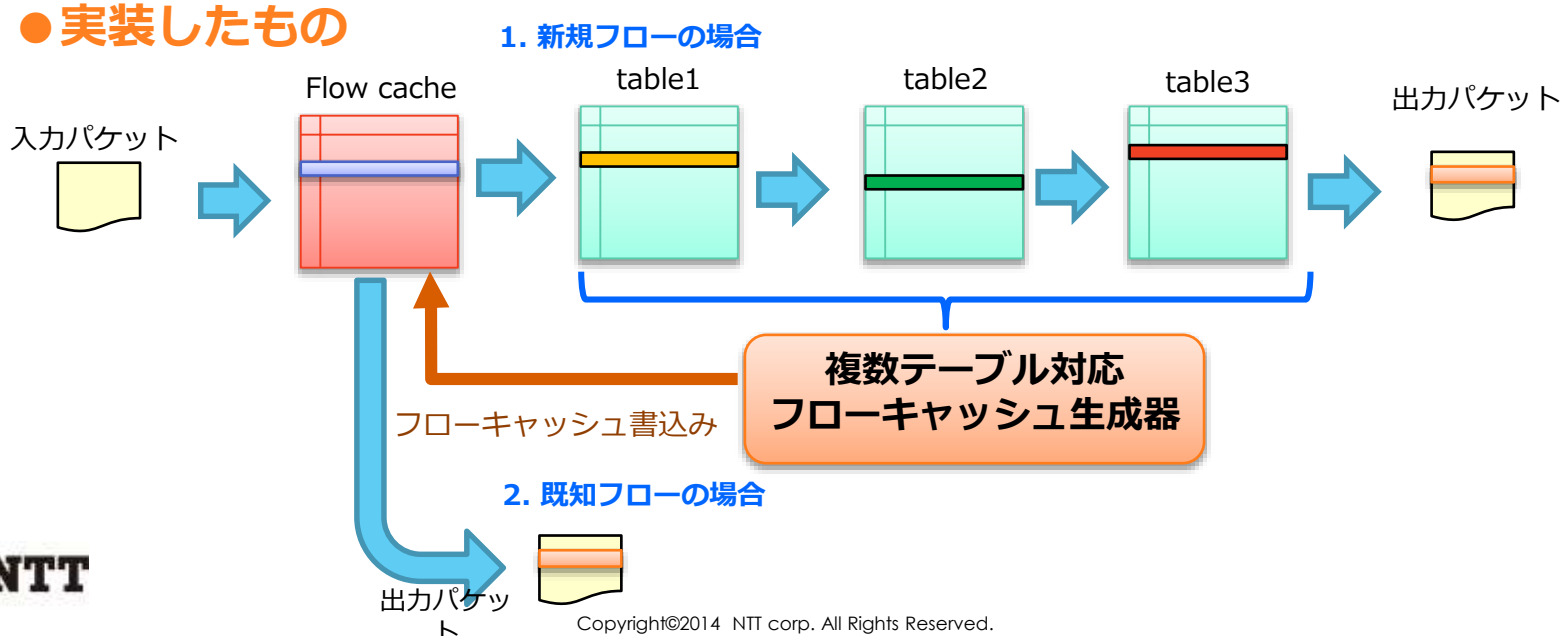
■ 毎回検索するのは厳しい

- とくに複数テーブル設定時
- あたったものだけキャッシュ

●シンプルな実装



●実装したもの



35

Lagopus評価

性能評価

■ まとめ

- Throughput: **10Gbps wire-rate**
- Flow rules: **1M flow rules**

  4000 flowmod add / sec

■ 評価モデル

- WAN-DC gateway
  - MPLS-VLAN mapping
- L2 switch
  - Mac address switching

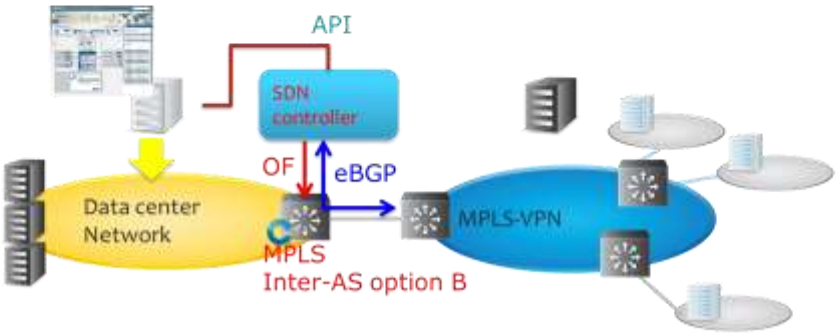
Evaluation scenario

Usecase : Cloud-VPN gateway

From ONS2014 NTT COM Ito-san presentation

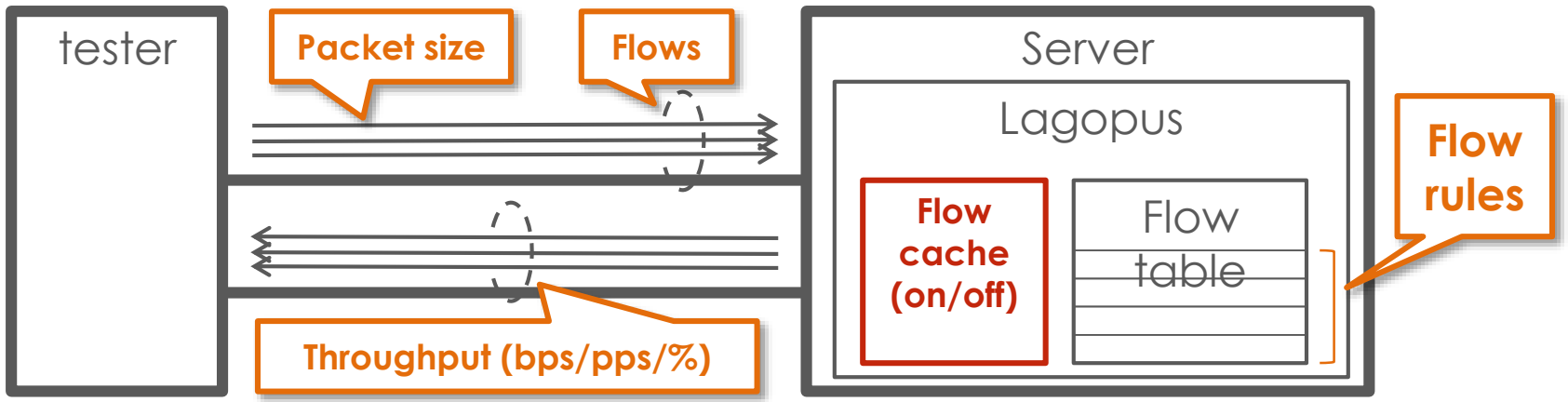
Tomorrow :

- Automatic connection setup via North Bound APIs
- SDN controller maintain mapping between tenant logical network and VPN
- Routes are advertised via eBGP, no need to configure ASBRs on provider side



性能評価

環境

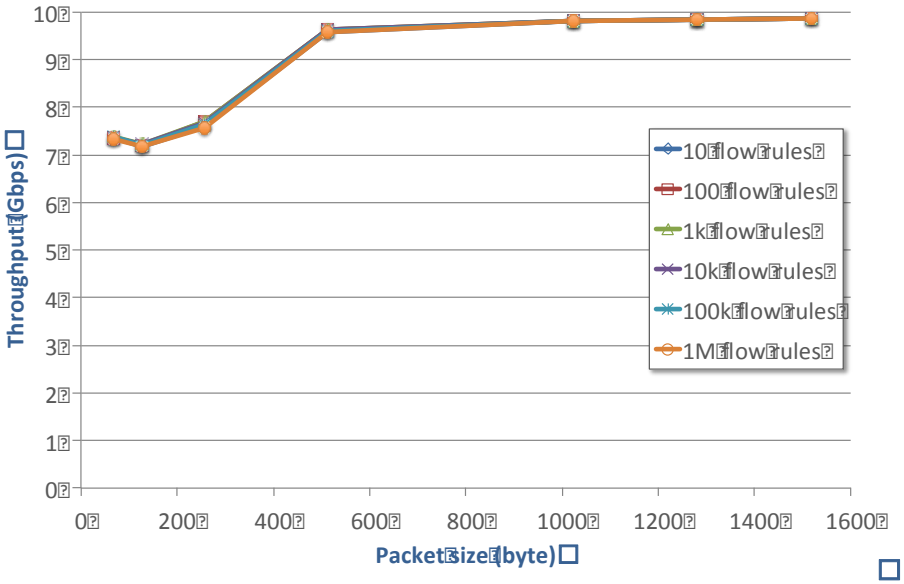


Server spec.

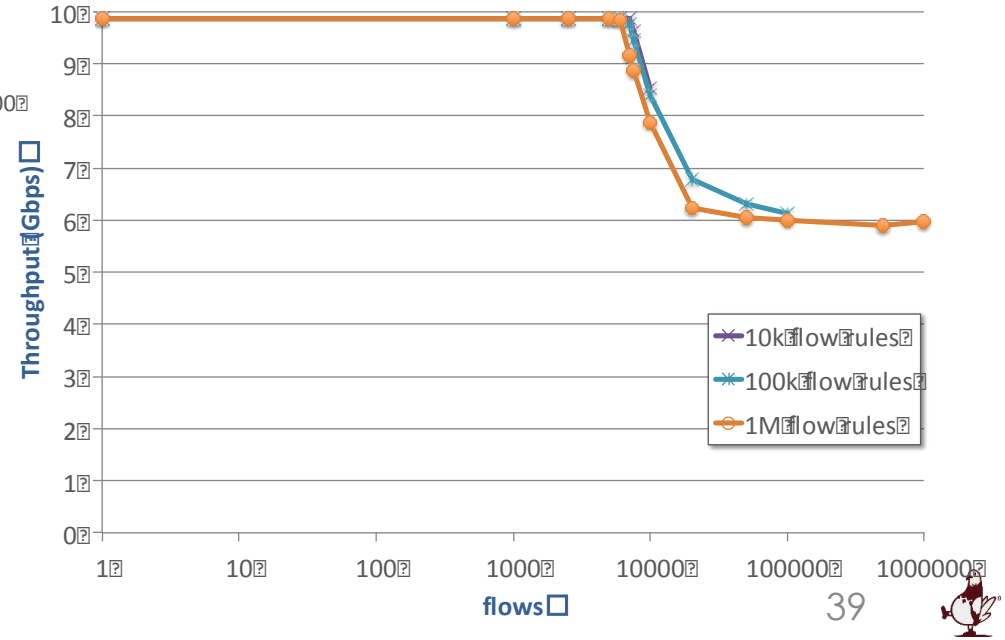
- CPU: Dual Intel Xeon E5-2660
  - 8 core(16 thread), 20M Cache, 2.2 GHz, 8.00GT/s QPI, Sandy bridge
- Memory: DDR3-1600 ECC 64GB
  - Quad-channel 8x8GB
- Chipset: Intel C602
- NIC: Intel Ethernet Converged Network Adapter X520-DA2
  - Intel 82599ES, PCIe v2.0

WAN-DC Gateway

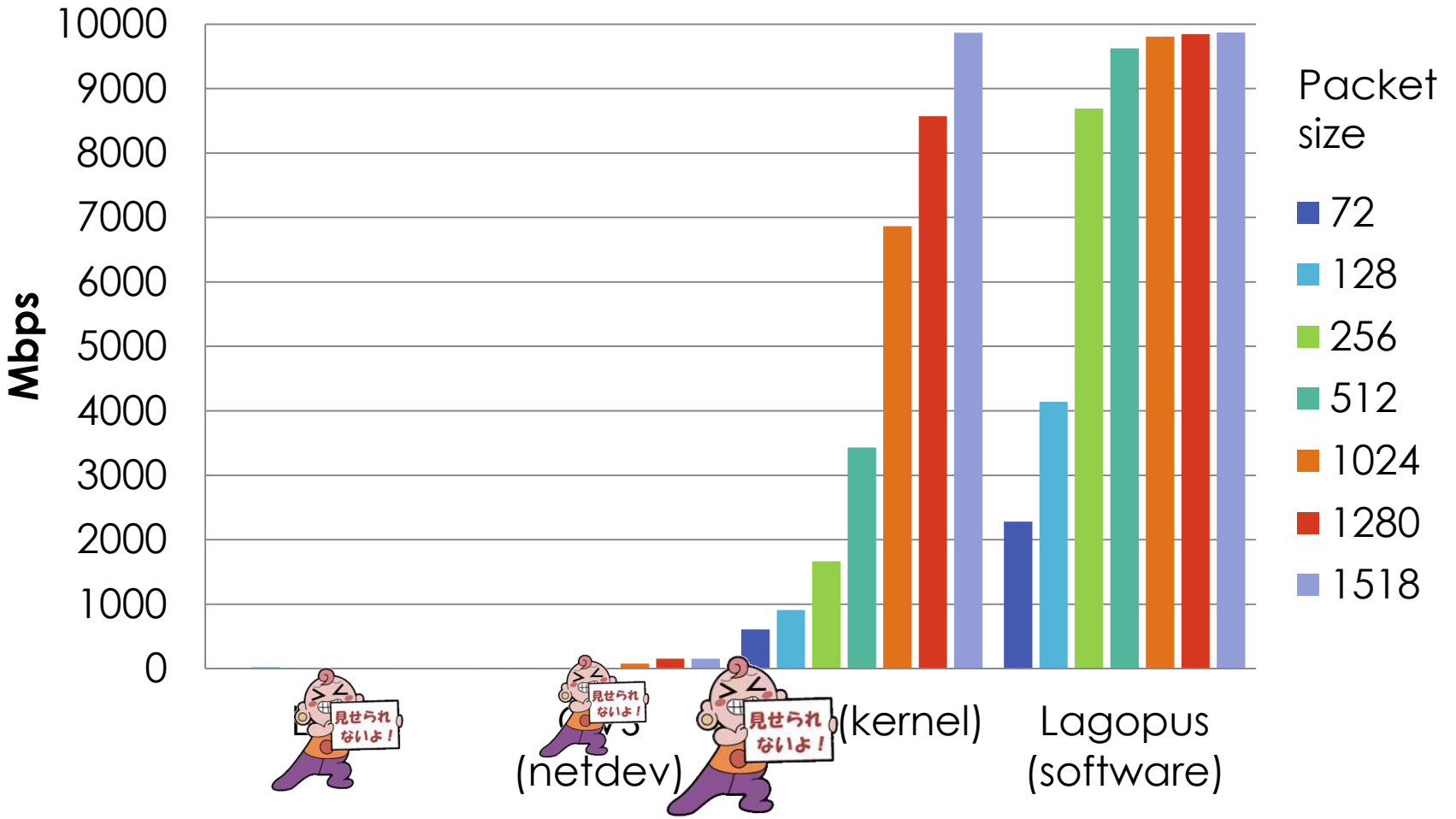
Throughput vs packet size, 1 flow, flow-cache



Throughput vs flows, 1518 bytes packet



L2 switch performance (Mbps)
10GbE x 2 (RFC2889 test)



まとめ

- 複雑なことをすると負け
- メモリコピーしたら負け
- ロックしたら負け
- シンプル is Best

- Lagopusは今後も高速化のための拡張を継続していきます！

Open Source Activity

- **Okinwa Open Days 2014にてハンズオン**
  - OpenFlow関連Ryu + Lagopusチュートリアル
- **Internet Week 2014 (11/18)にてBoF**

http://Lagopus.github.io/
Twitter: lagopusvswitch