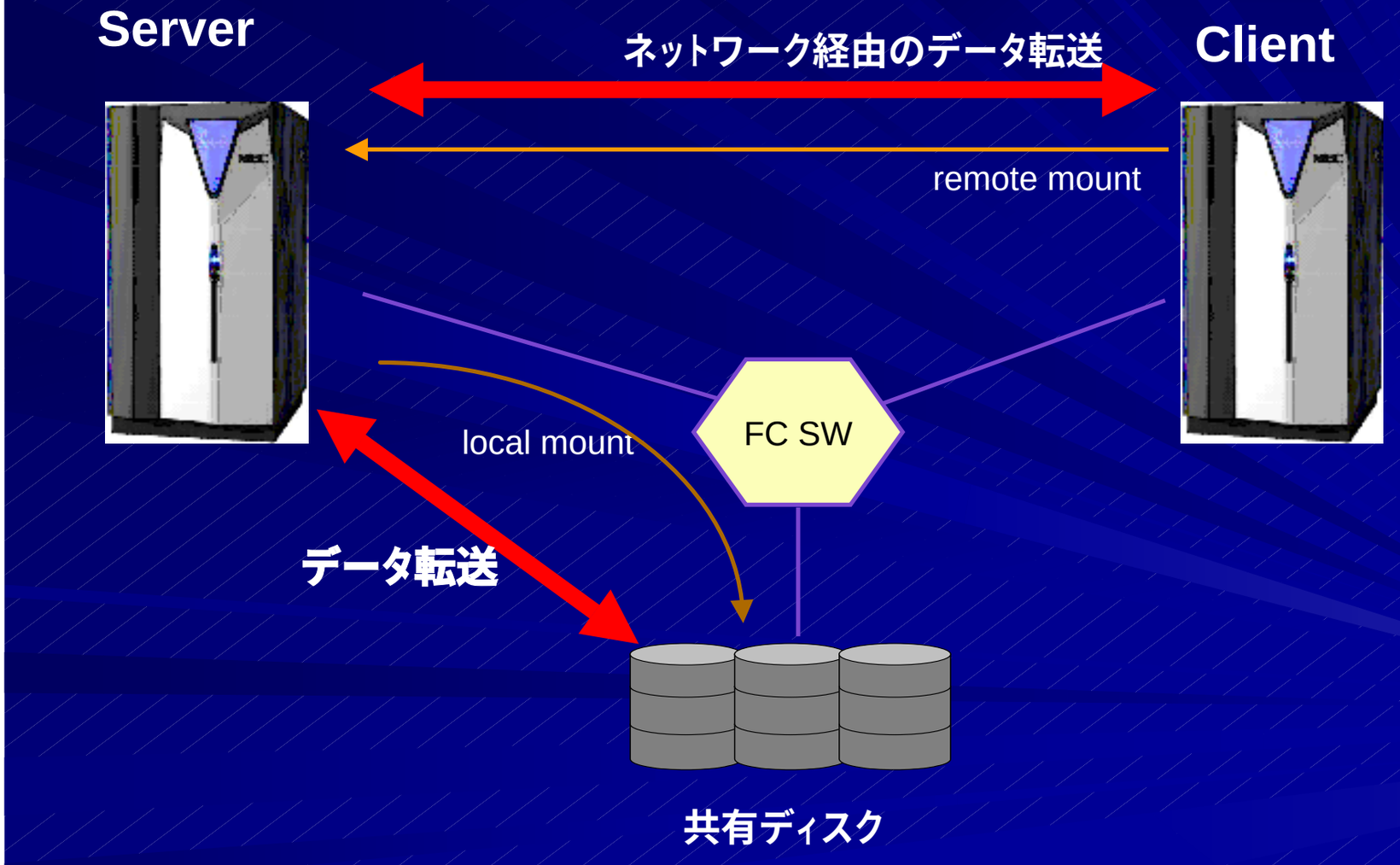
クラスタシステムにおける高速ファイル共有のための一手法

大谷敦久 青野寛 外丸浩子 (NEC)
佐竹康司 (NECソフトウェア東北)

発表内容

- 背景
  - 高速ファイル共有とNFSとの違い
  - サードパーティー転送による手法
- 目的
- デーモン方式による手法
- ストライピング時の処理
- 性能検証
- まとめ
- 今後の課題

NFSによるデータ転送の経路

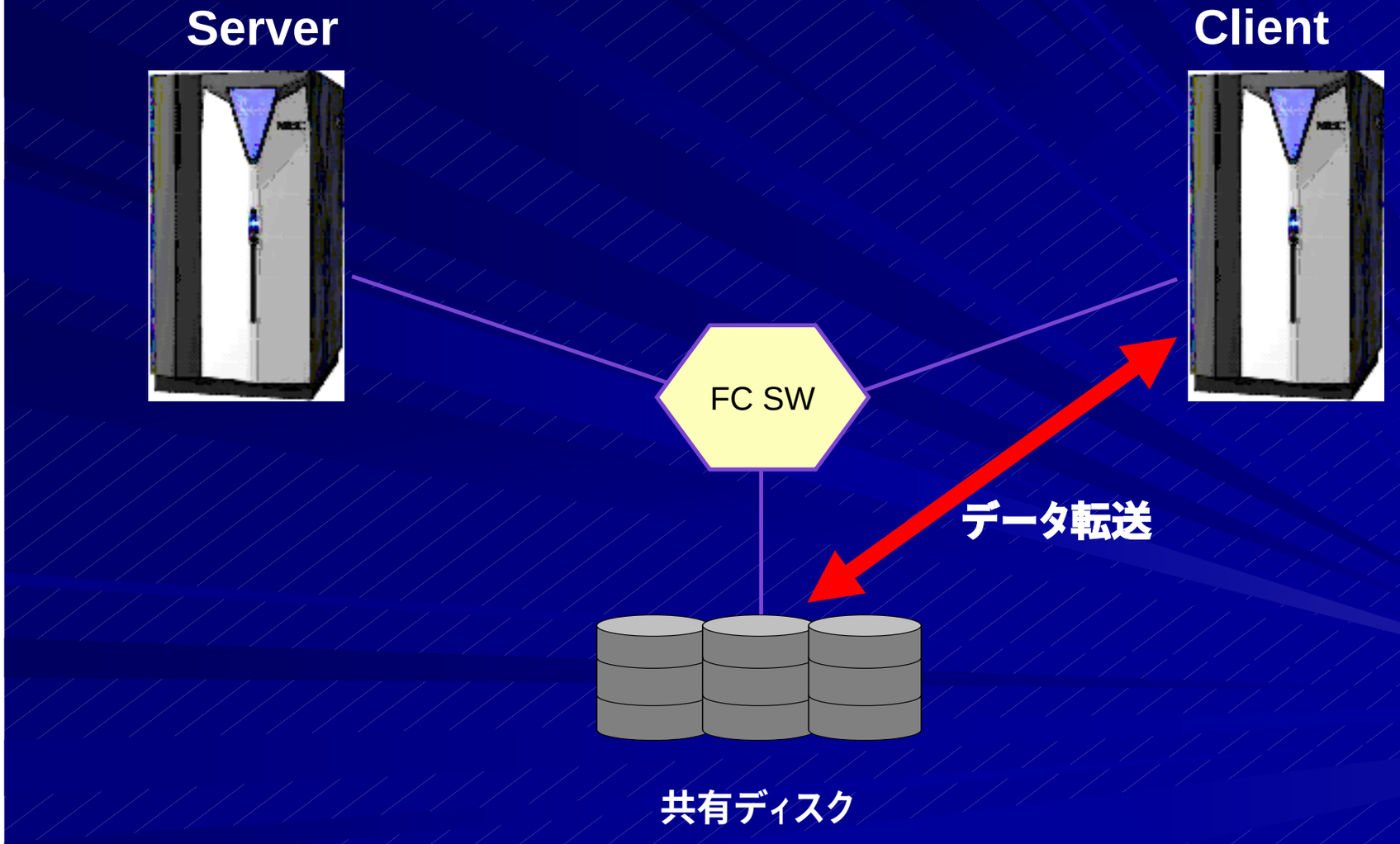


NFSの問題点

- クライアントとサーバー間、サーバーとディスク間の2度データ転送が必要。
- データが細切れになる(4KB, 8KB 程度)

NFSは、ディスク装置本来の性能を発揮できない。

高速ファイル共有によるデータ転送の経路

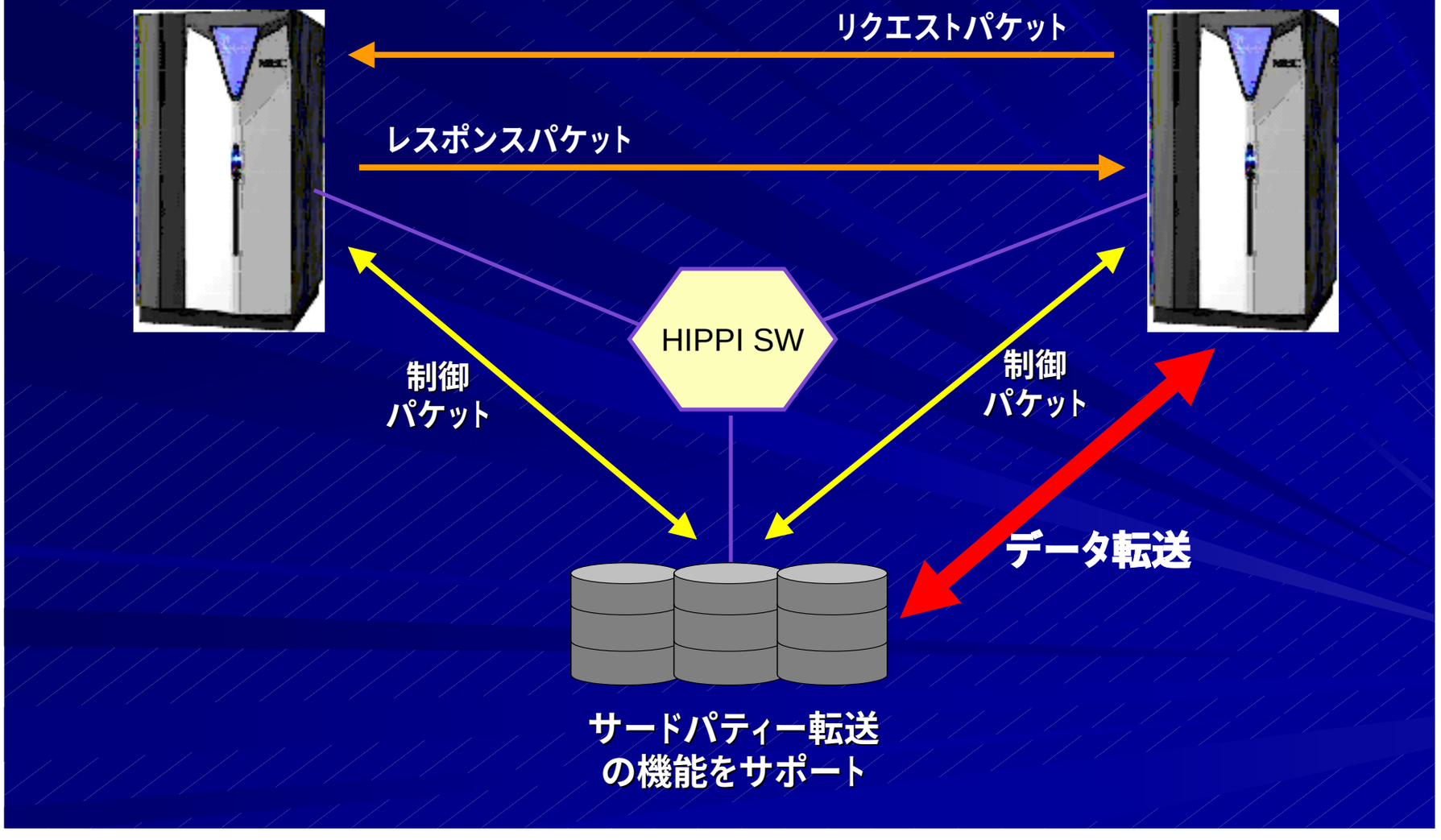


ファイルシステムの整合性の維持

- ファイルシステムの情報はサーバーが保持
  - クライアントは、ファイルシステムの情報を持たない。
    - 対象ファイルのディスク上の物理的な位置情報を知らない。

サーバーは、クライアントに対して、
この制御情報の送信が必要

サードパーティー転送に基づくファイル共有(TPT方式)



TPT方式の問題点

- サードパーティー転送の機能をサポートしたディスク装置が必要
  - 価格や性能の面で、好みのディスクを選べない。
- Fibre Channel では、実現困難
  - FC の HBA は、通常イニシエータとしての機能しかサポートしていない。

サードパーティー転送を使用しない方式が必要

ストライピングについて

- HPC関連分野では、ストライピングによりI/O性能を向上させることが一般化。

SANにおけるファイル共有においても、
ストライピングのスケーラビリティーの維持が必要

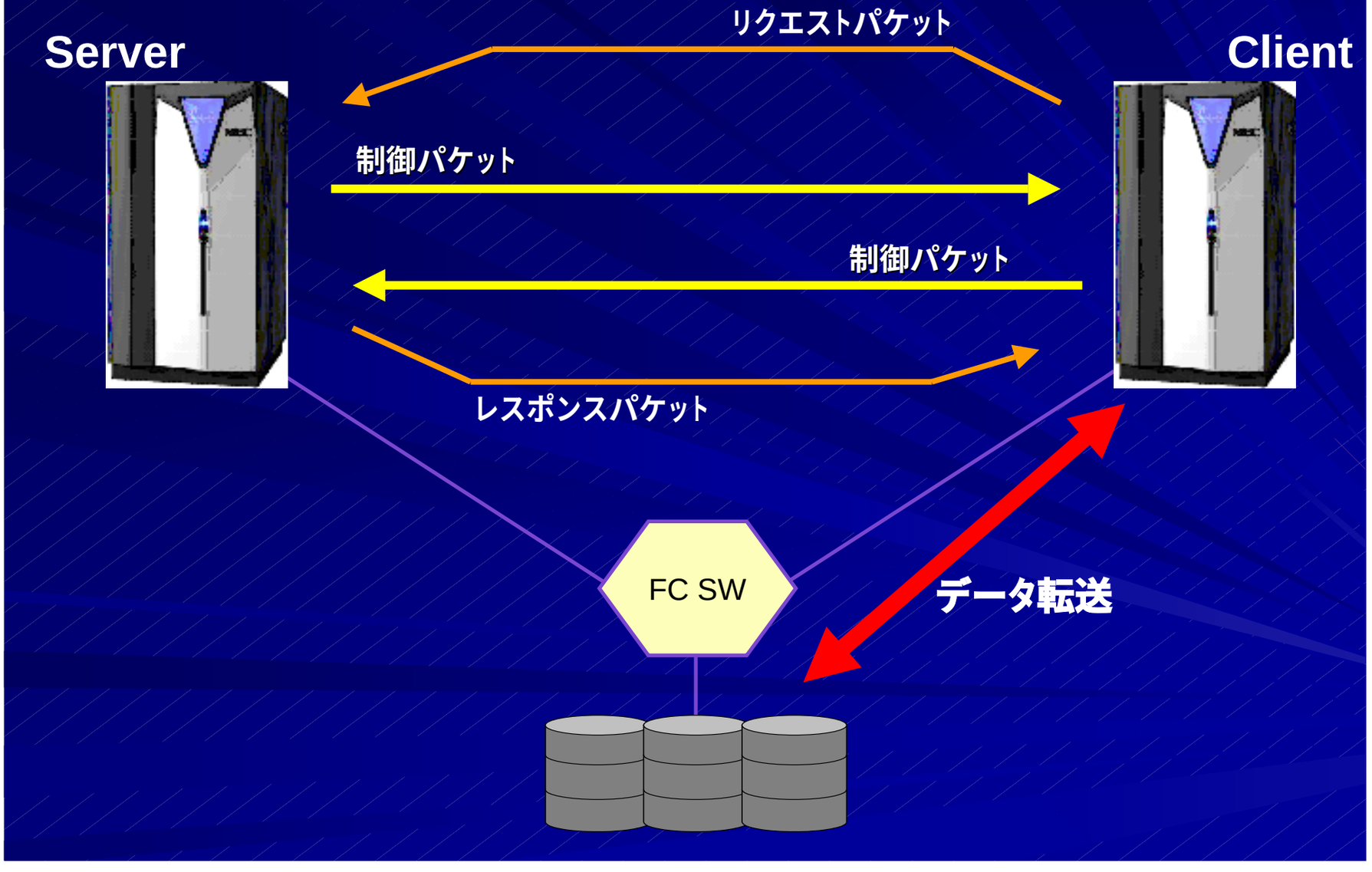
目的

- サードパーティー転送を使用しないファイル共有の手法の開発。
- ローカルI/Oに匹敵するI/O性能が得られる。
- ストライピングした場合でも、I/Oのスケーラビリティーを低下させない。

サードパーティー転送を使用しない方法

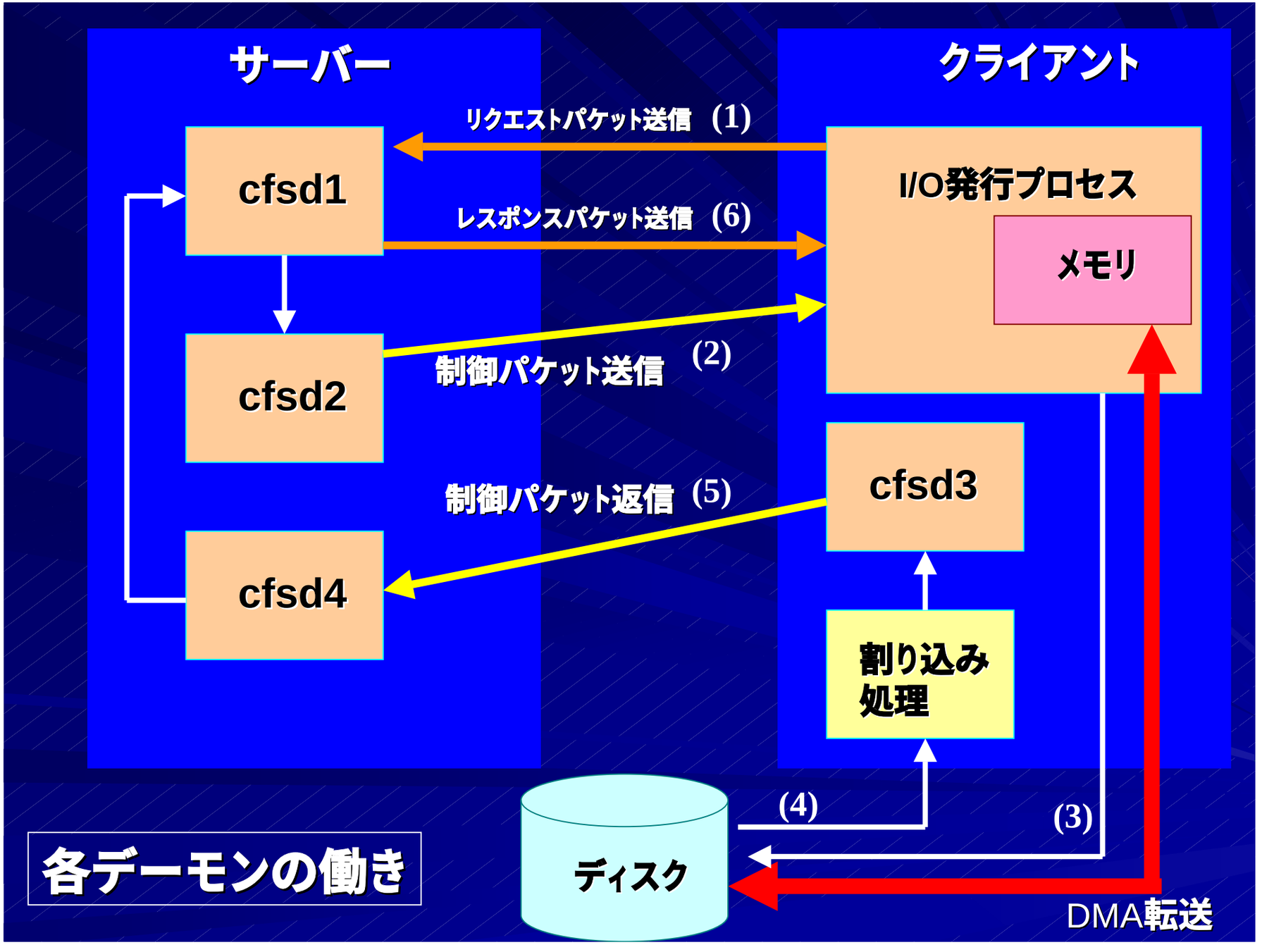
- ネットワークを経由して、制御情報を送信
  - 制御情報は、データサイズが小さいので、ネットワークを経由してもほとんどI/O性能に影響しない。

サードパーティー転送を使用しない方法



デーモン方式

- カーネルデーモンを用いて、制御情報の送受信をネットワーク経由で行う
  - サーバー
    - cfsd1、cfsd2、cfsd4
  - クライアント
    - cfsd3


サーバー
クライアント
リクエストパケット送信 (1)
cfsd1
I/O発行プロセス
レスポンスパケット送信 (6)
メモリ
cfsd2
制御パケット送信 (2)
cfsd3
制御パケット返信 (5)
cfsd4
割り込み
処理
(4)
(3)
ディスク
DMA転送
各デーモンの働き

リクエストパケットの内容

- ファイルハンドラ
- I/Oサイズ
- ファイルオフセット
- Read/Write
- 認証情報
- I/O識別子(Transfer-ID)

制御パケットの内容

- ディスク装置のスペシャルファイル名
- I/Oを開始するセクタ番号
- セクタ数
- I/O発行プロセスのI/O開始メモリアドレスからのオフセット
- Transfer-ID

制御パケットの内容(返信用)

- ディスク装置のスペシャルファイル名
- I/O発行プロセスのI/O開始メモリアドレスからのオフセット
- エラー情報
- Transfer-ID

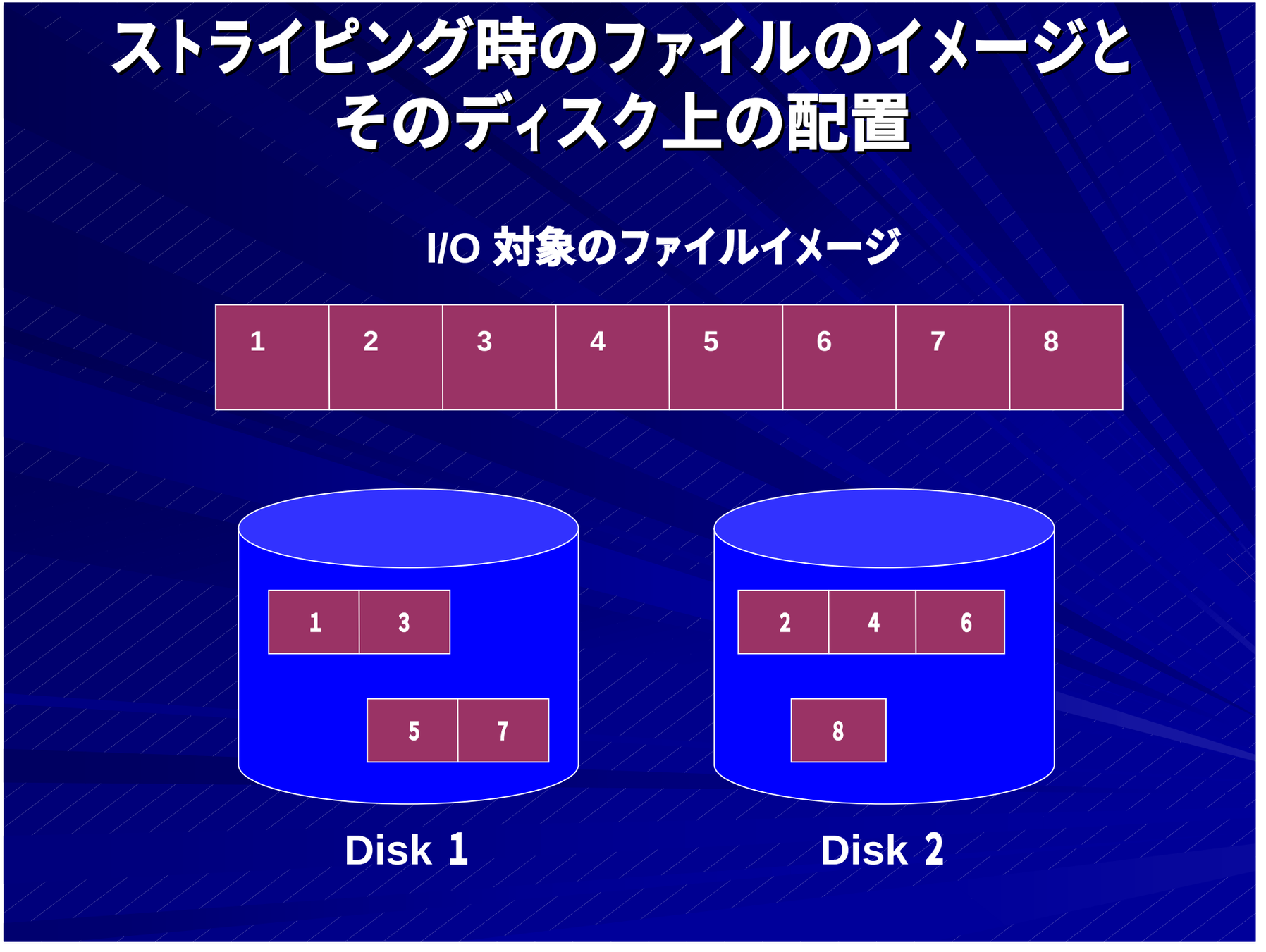
レスポンスパケットの内容

- 実際にI/Oできたサイズ
- エラー情報
- ファイルの更新時刻など
- Transfer-ID

ストライピング

- 1つのI/O要求に対して、複数台数のディスク装置へのアクセスが必要。
  - 並列にアクセスできなければ、ストライピングの意味がない。

ストライピング時のファイルのイメージとそのディスク上の配置



ストライピング時の問題点

- ストライピングしない場合
  - 物理的に連続したブロック毎に制御パケットをサーバーからクライアントへ送ることができる。
- ストライピングした場合
  - ストライピングサイズ毎に異なるディスク装置へI/Oを発行すると、I/O発行の単位が細分化される。

制御パケットの転送量が膨大になり、ネットワークトラフィックが増大する

パケット転送回数の軽減

- ディスク内の連続領域毎にまとめてI/Oを発行。
- 同一ディスク毎に制御パケットを1つにまとめる。
  - 「連続領域」と「同一ディスクの別領域」の2次元構造とする。

Disk 1 への制御パケット

1	3
5	7

Disk 2 への制御パケット

2	4	6
8		

上記例では、8回パケット送信が必要なところを2回に軽減できる。

クライアントでのI/O処理の効率化

- ストライピング数分の制御パケット受信が必要
  - クライアントでは、連続的なI/O発行が必要

発行したI/Oの終了を待ち合わせていると、並列性能が出せない。

クライアントでのI/O処理の効率化(続き)

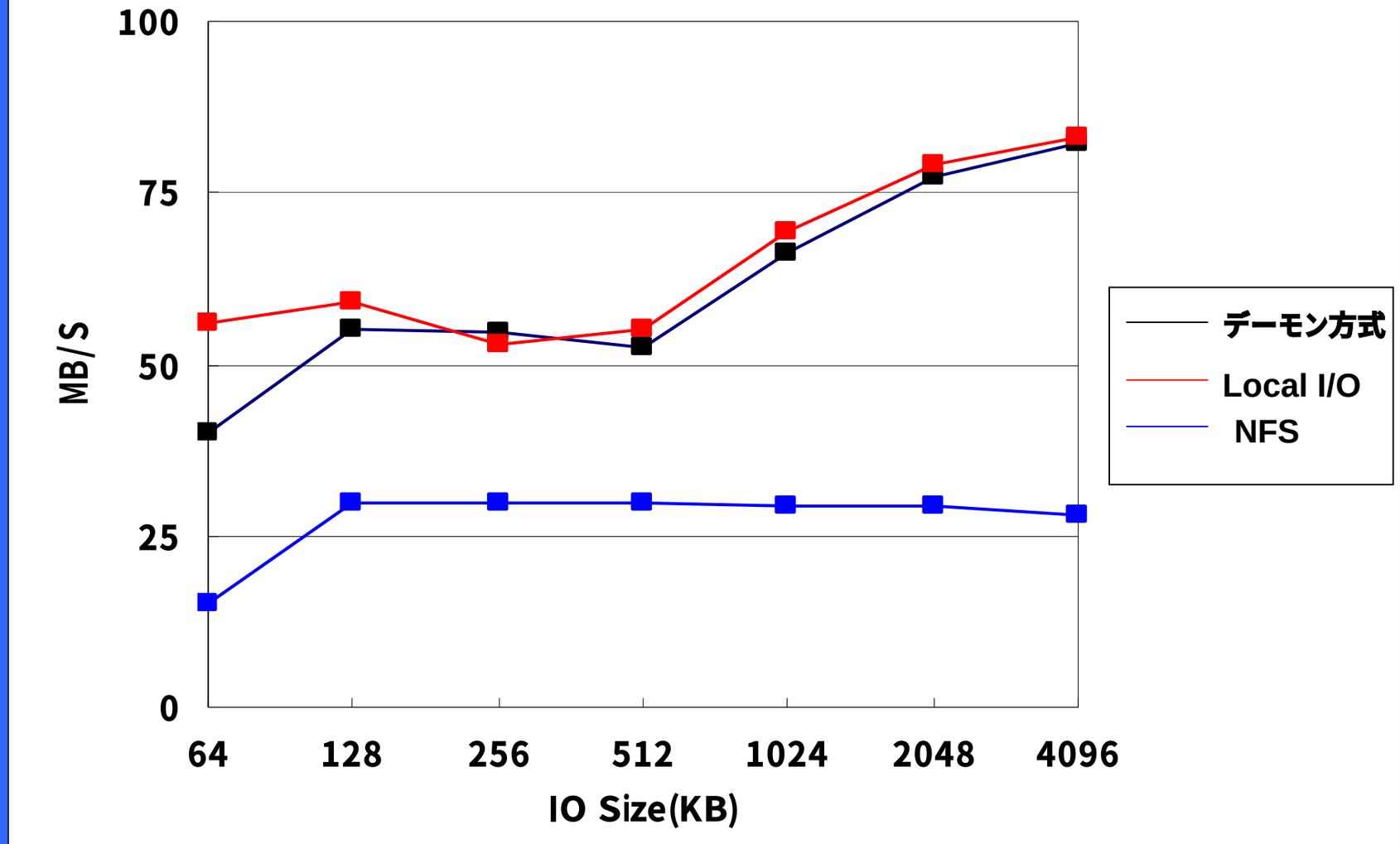
- 「I/O発行処理」と「I/O終了処理」の分離
  - クライアントへの制御パケットの送信からI/Oの発行
  - I/O終了通知からサーバーへの制御パケットの返信

「I/O発行処理」と「I/O終了処理」は、
並列に動けることが必要。

性能測定結果

- 測定環境
  - NEC Express5800/1160Xa（2台：サーバー、クライアント）
  - Fibre Channel アレイディスク装置 (4+P、RAID5)
  - ネットワーク：Gigabit Ether
  - ベースカーネル：2.4.17、ファイルシステム：SGI XFS
- 測定項目
  - ローカルI/O、NFSとの比較
  - ストライピングのスケーラビリティー
  - パケット転送回数の低減効果の検証
  - 複数のI/Oを同時に発行した場合のスケーラビリティー

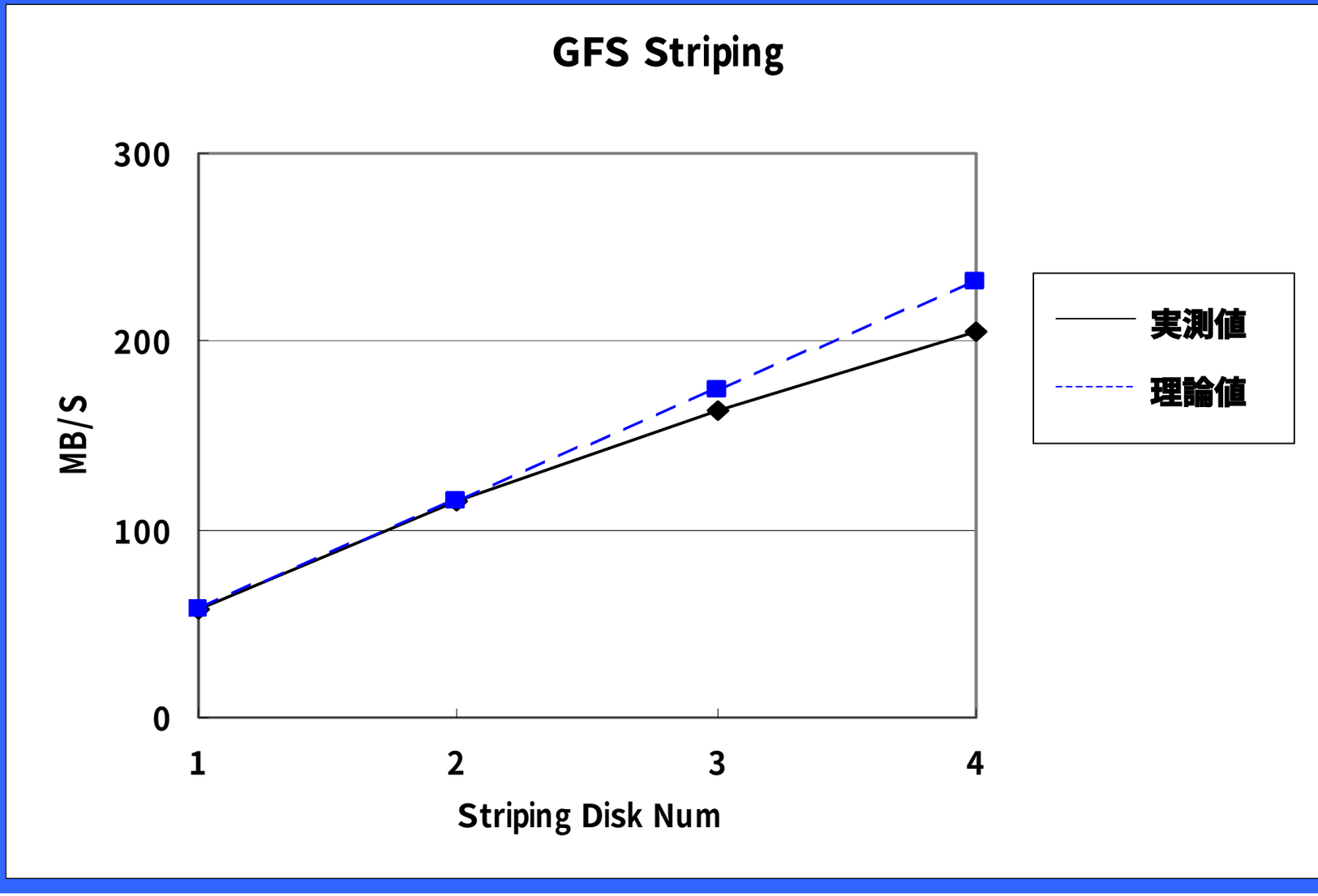
ローカルI/O、NFSとの性能比較



ストライピングのスケーラビリティー
理論値との比較

- LVM (Logical Volume Manager)を使用
- ストライピングサイズ：64KB
- ストライピング数：1～4
- I/Oサイズ：2MB

ストライピングのスケーラビリティー
理論値との比較

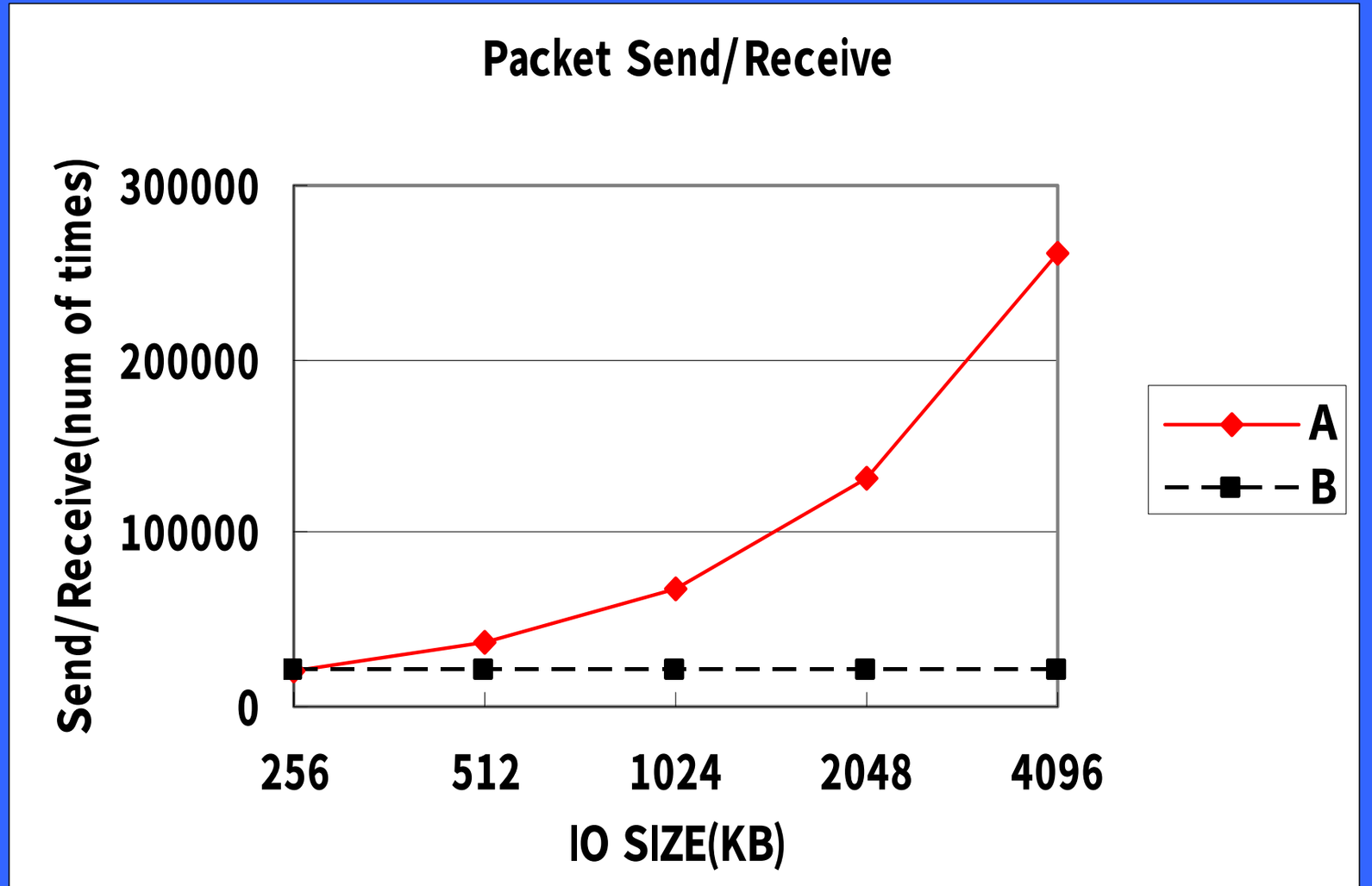


パケット転送回数軽減の効果

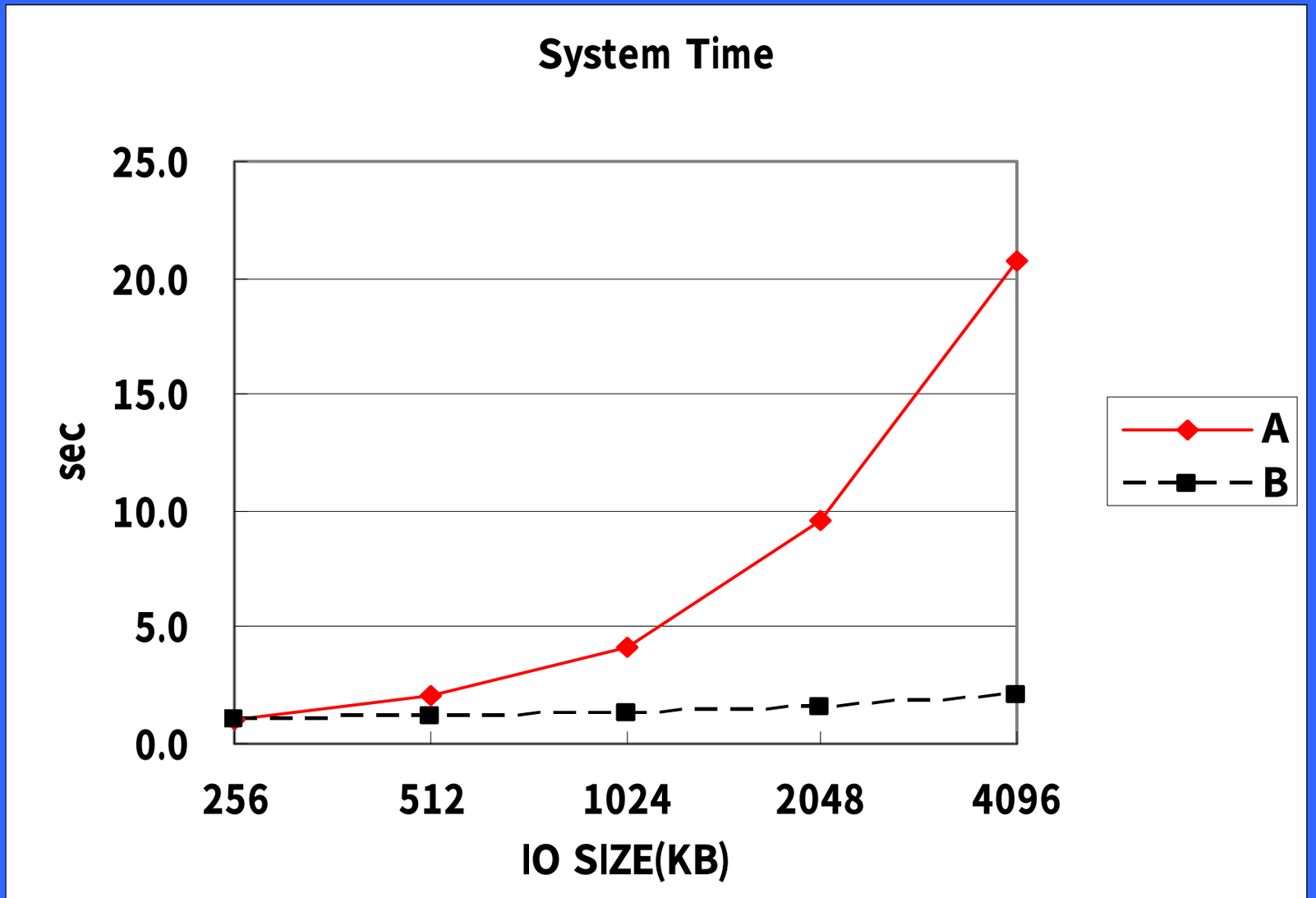
- 方法A：ストライピングサイズ毎に別々に転送(比較対象)
- 方法B：パケットをまとめて転送
- ストライピングサイズ：64KB、ストライピング数：4
- I/Oサイズ：256KB~4MB

- 上記について、以下を比較
  - パケット転送回数
  - I/O発行プロセスのシステムタイム

パケットの転送回数の比較



システムタイムの比較

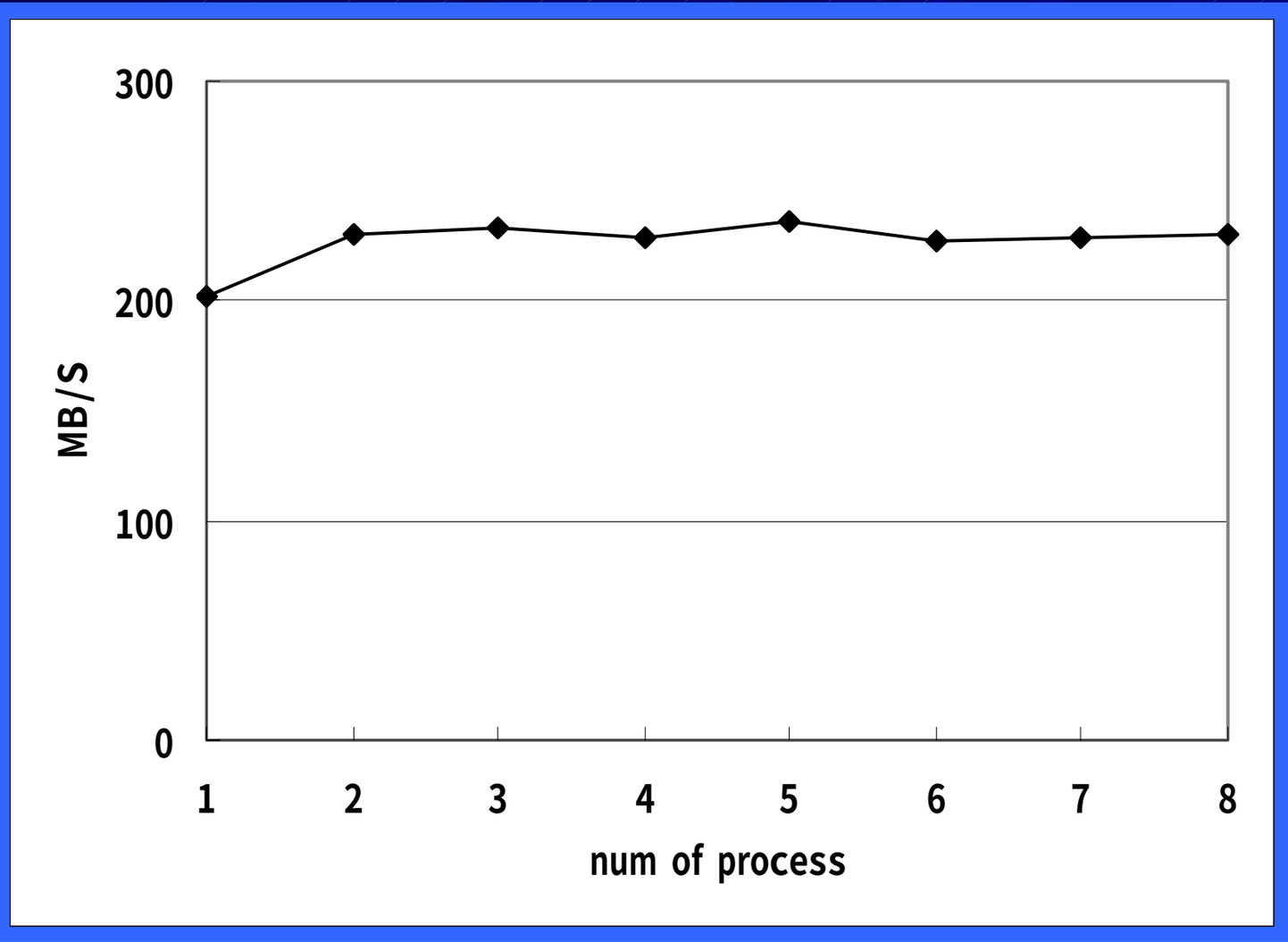

System Time
25.0
20.0
15.0
10.0
5.0
0.0
sec
256
512
1024
2048
4096
IO SIZE(KB)
A
B

複数I/Oの同時発行のスケーラビリティー

- 理論上は、
  1プロセスの性能値＝多プロセスの性能の合計値

1～8プロセスの場合について、性能測定を実施し、上記を検証

複数I/Oの同時発行のスケーラビリティー



まとめ

- 本手法は、サードパーティー転送をサポートしないディスク装置でも適用可能。
- 本手法の性能は、128KB以上ではローカルI/Oの90%以上であり、ローカルI/Oに近い性能が得られる。
- ストライピング時の本手法のスケーラビリティーは、4ストライピングにおいて理論値の88%である。
- サーバーにおいて、制御情報を1つのパケットにまとめることで、ストライピング時の処理の効率化ができる。

まとめ(続き)

- 複数のI/Oを同時に発行した場合でも、処理効率は低下しない。

今後の課題

- 多数のディスクでストライピングする場合、I/Oロックの細分化が必要であると思われる。

- カーネル2.5におけるI/Oロックの細分化に対応し、その効果を検証する。