トポロジを考慮するデータ転送スケジュラー

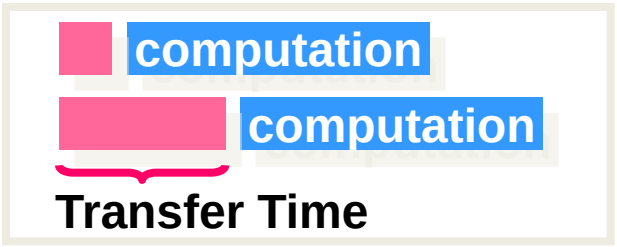
高橋 慧　田浦健次朗　近山 隆

データ転送計画の重要性

- データ・インテンシブなアプリケーションが分散環境で実行される機会が増加している
  - 大量のデータを処理することにより、従来得ることの出来なかった成果を上げている
  - Web上のデータの分析・自然言語処理など
- ノード間でのデータ転送計画が重要になる
  - 転送に無視できない割合の時間を占める
  - 分散環境ではバンド幅が不均質なので、計画によって転送時間が大きく変化する

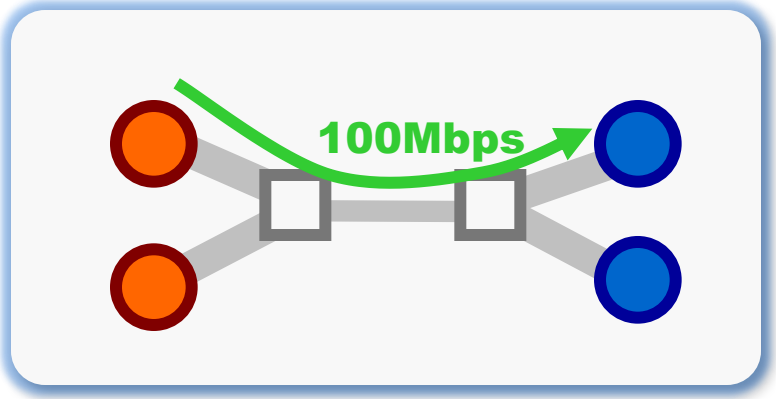
例: データ転送計画の重要性

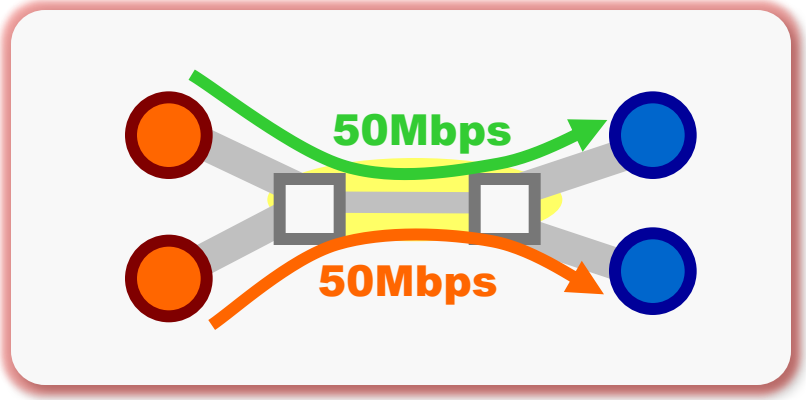
- クローラが集めたページが様々なノードに分散しているとき、これを多数のノードを用いてパージングする
  - 一般的なバンド幅: 1Gbps-10Mbps
  - Chasenパーザのスループット: 1.7MB/s = 14Mbps
- 転送計画によって処理時間が20%も変化する
  - 10Mbpsのリンクを複数の転送が共有するとさらに転送速度が低下する



リンク共有による影響

- 複数の転送がリンクを共有すると、転送速度の合計は一定値(バンド幅)で制約される
  - $\Sigma_{i \in \text{全ての転送}}$ (転送iの速度) ≤ (リンクのバンド幅)
  - 4つの転送があれば、転送速度は25%に低下





トポロジ情報の利用

- ネットワーク・トポロジの情報を用いると、リンクを共有しない転送計画を立てられる
  - トポロジは高速(≈15sec)で推定可能[1]

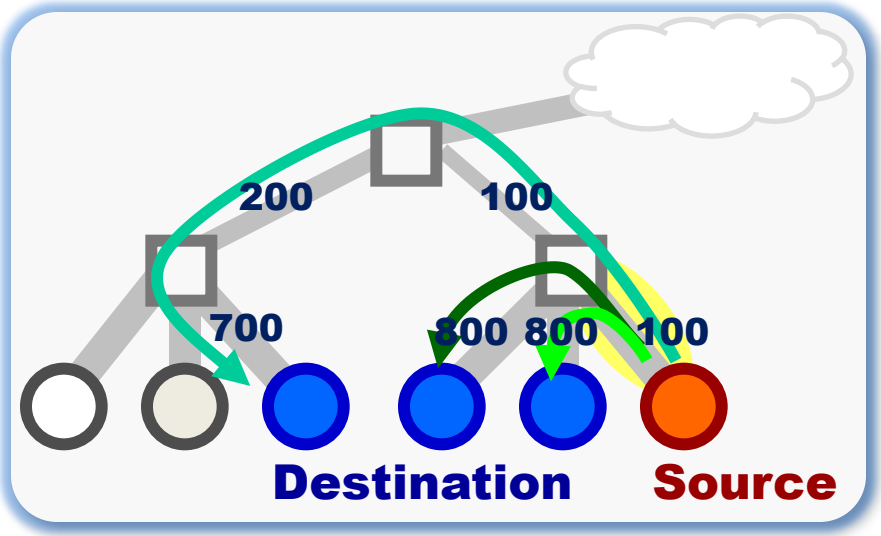
- トポロジ情報を利用し、1ソースからのマルチキャストを効率よく行う例を次に示す

[1] Shirai et al. A Fast Topology Inference --- A building block for network-aware parallel computing. (HPDC 2007)

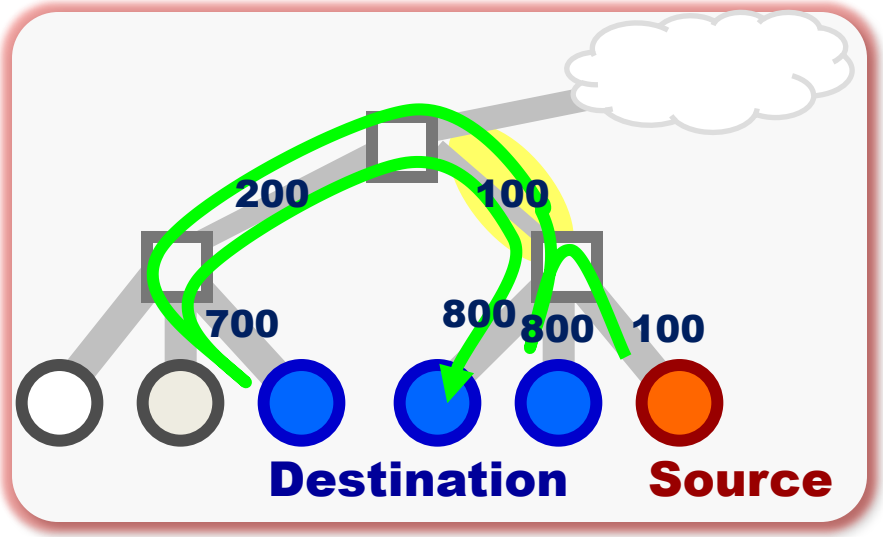
1ノードからのマルチキャスト

- あるノード(source)から複数のノード(destination)に大きなデータを送信する
  - Sourceが全destinationに直接転送すると非効率
  - ランダムにパイプラインを作ると多くのリンク共有が発生

3転送が100のリンクを共有

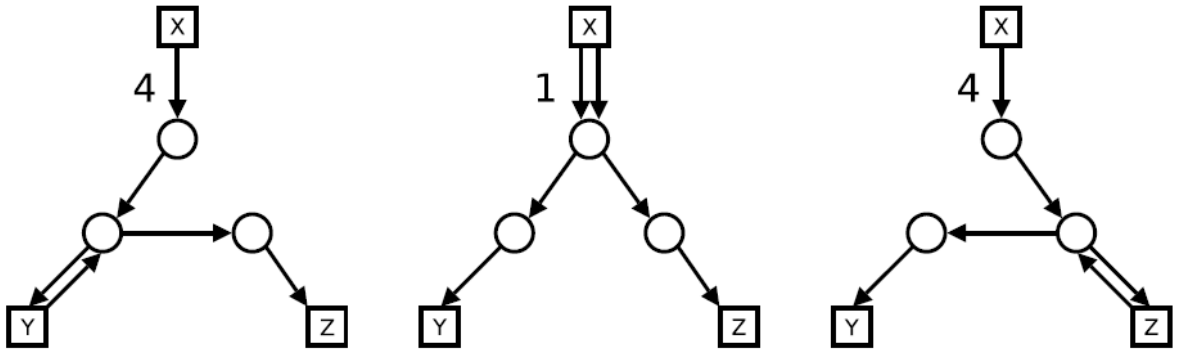


2転送が100のリンクを共有



Balanced Multicast

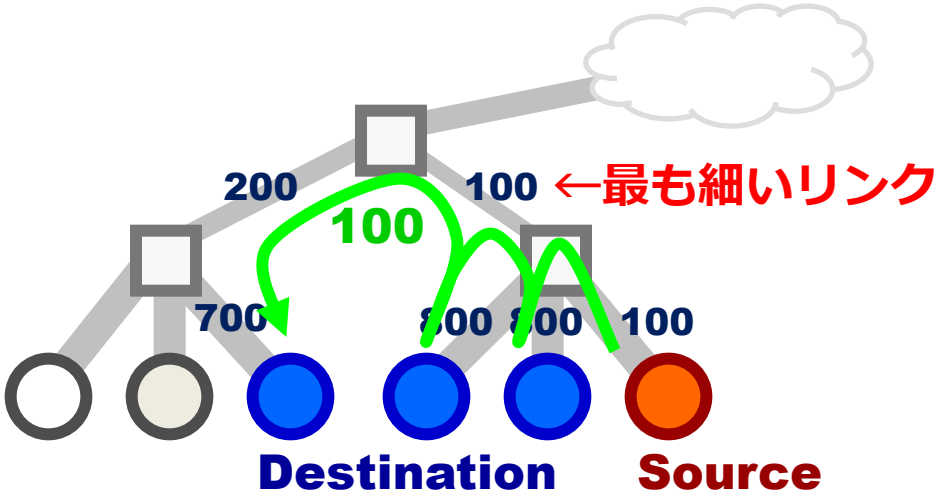
- [1]Burgerらは、クラスタ間のトポロジ及びバンド幅を考慮し、効率の良いマルチキャストを計画するアルゴリズムを示している
  - 様々なツリーの候補を生成
  - 線形計画法で重みを与え、重ね合わせる
- クラスタ内のリンクの共有は考慮されていない
  - 考慮すると計算量がとても多くなってしまう



[1] Burger et al. Balanced Multicasting: High-throughput Communication for Grid Applications (SC2005)

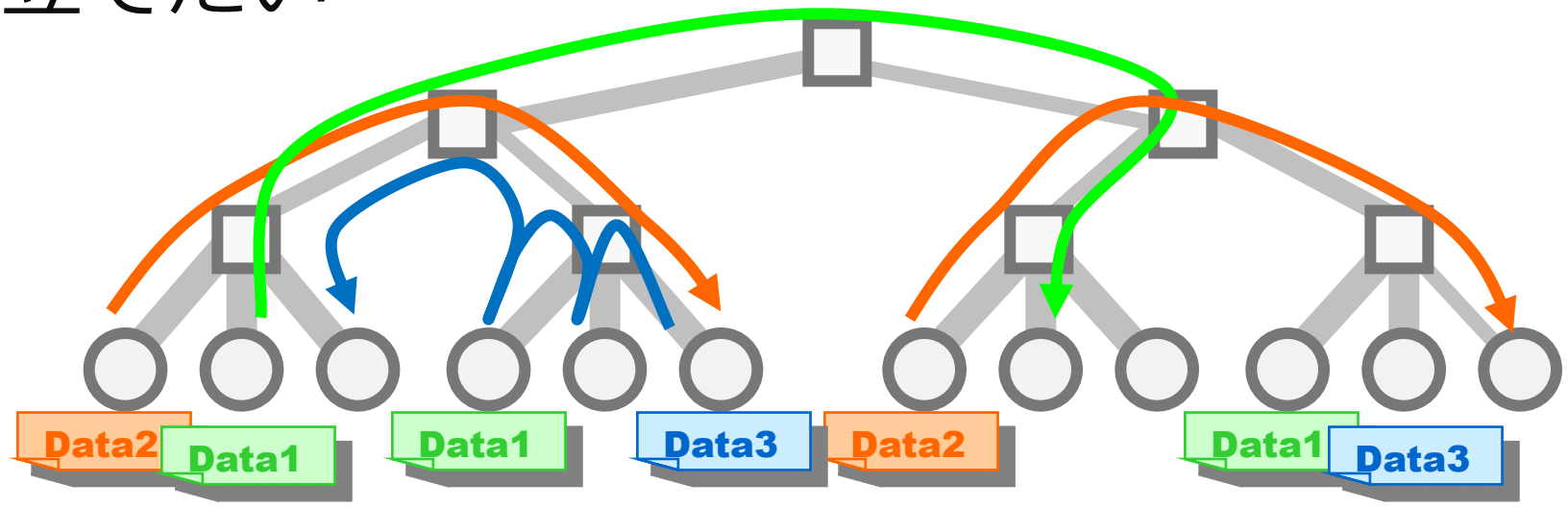
パイプライン・マルチキャスト

- トポロジを幅優先に辿ってパイプライン転送
  - ツリーの場合、全ノードが同期される時間が最短となる
  - 各リンクを一方向につき一回ずつしか使わないため、リンクの共有が起こらない:
- 転送時間は (size/min(bandwidth)) + RTT * Nとなる
  - データが大きい場合 ノード数Nに対しO(1)
  - 転送速度はパイプライン中の最も細いリンクに制約される



もっと複雑な転送問題

- 実際の環境では:
  - 多種類のデータの転送が同時に発生する
  - 以前転送したデータを再び転送する場合、転送元に複数の候補がある (複数のソースがある)
- トポロジ情報を用いて、効率的な転送計画を立てたい



本研究の貢献

- トポロジを用い、分散環境で複数のデータ転送を効率的に行うアルゴリズムを提案する
- 具体的には、

1. 複数の転送があり、
2. それぞれの転送が複数のソースを持つとき、
3. 各転送先に到着するスループットの合計が最大になるような、

転送計画を立てるアルゴリズムを提案する。

発表の流れ

- はじめに
- 問題設定
- アルゴリズム
- 実験・考察
- 終わりに

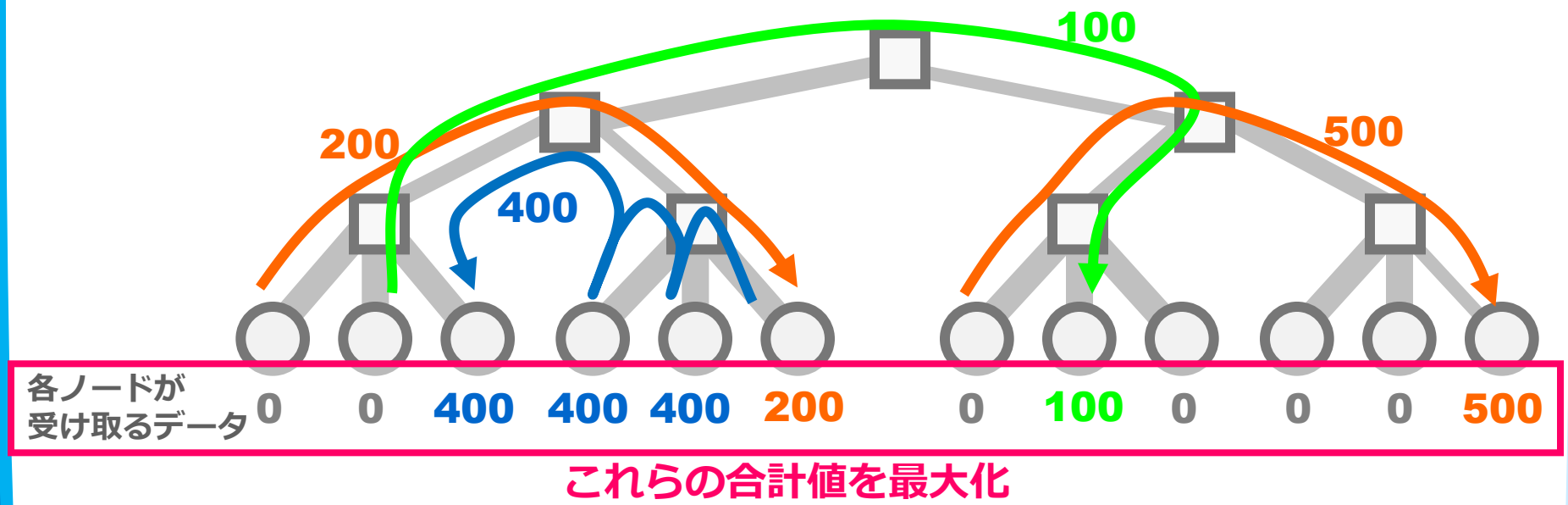
問題の定式化

- トポロジ・リンクのバンド幅が与えられる
  - リンクは上り下り別々に考える
  - 非対称でも良い
- N種類のデータ転送がある
  - 各転送は複数の転送元(ソース)と転送先を持つ

目標: スループット最大化

- 全転送について、各ノードが受け取るデータのスループットの合計値を最大化したい

$$\sum_{i \in for\_every\_destination} (arriving\_throughput(i))$$



最大スループット問題のNP完全性

- 最大スループット問題は1転送のみを考える場合でもNP完全
  - 3-SATに帰着できる (付録をご覧下さい)
  - 転送元がNs個, 転送先がNd個のとき、$O(Ns^{Nd})$
- 実際的な時間で解くためには、発見的手法を取る必要がある
  - 本研究では、「ボトルネックリンク最大化問題」を利用して最適解に近づけることを考える

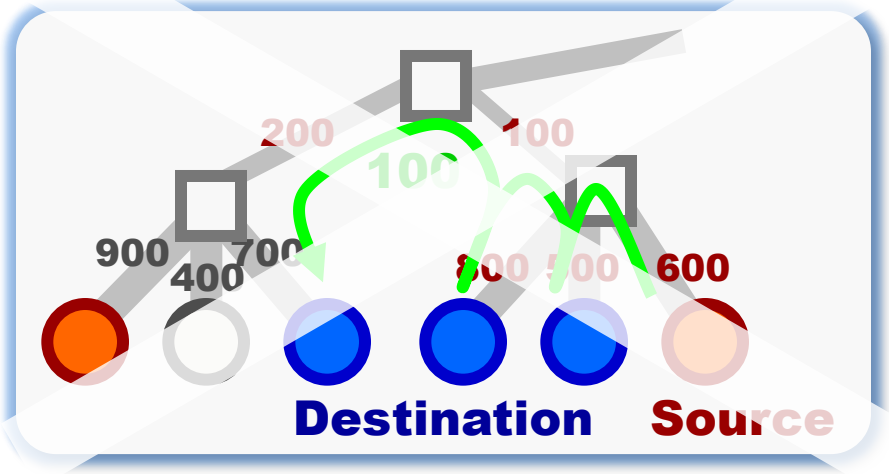
発表の流れ

- はじめに
- 問題設定
- アルゴリズム
  - より簡単な問題と、その解法
  - 提案するアルゴリズム
- 実験・考察
- 終わりに

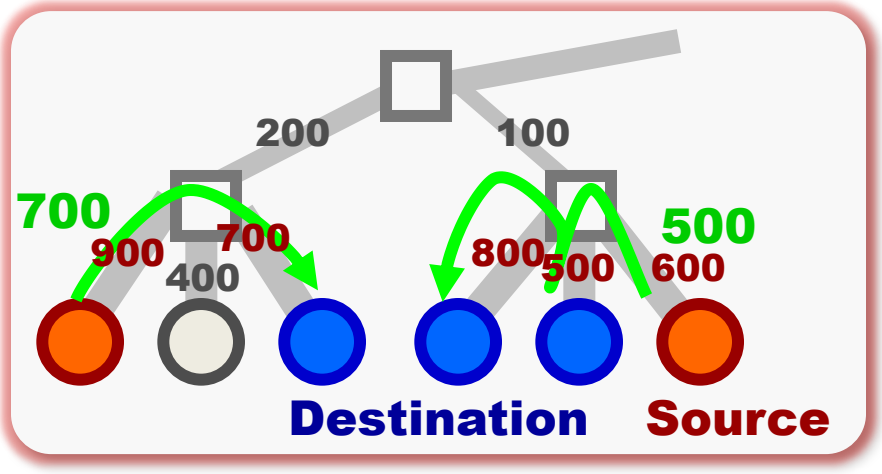
より簡単な問題: ボトルネック最大化

- 1種類のデータのマルチキャストにおいて、ボトルネック(最小のバンド幅)リンクを最大化
- 1種類のデータ・複数ソースの最大スループット問題の近似解が求まる
- Dynamic Programmingによって最適解が求まる

最小バンド幅 = 100



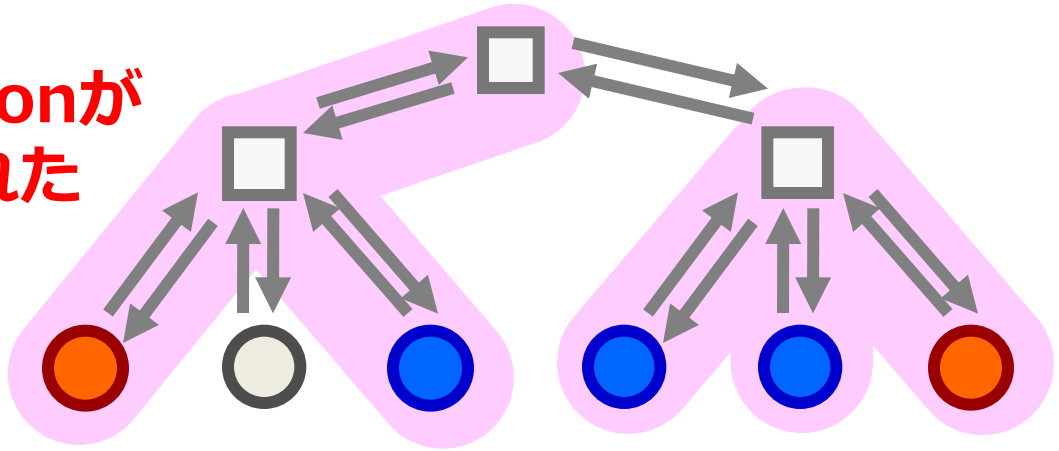
最小バンド幅 = 500



ボトルネック最大化のアルゴリズム (1)

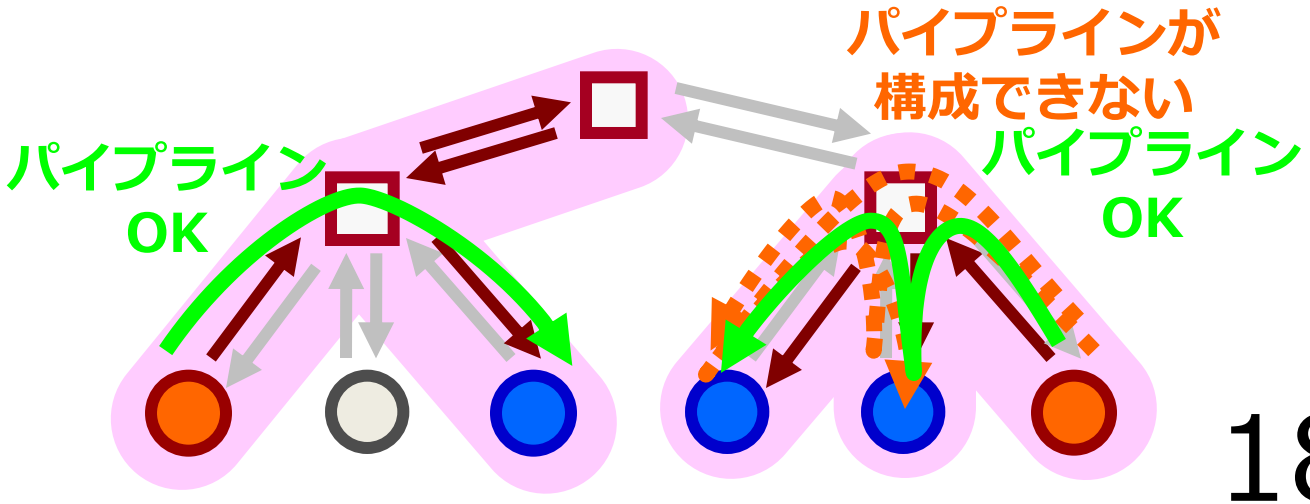
- 定義
  - A:ソースから到達できるノード集合
  - origin(l), end (l): リンクlの始点と終点
  - src(a): aに対し到達可能なソース
- 順番にリンクを追加
  - {l: origin(l)∈A} 中で最大のバンド幅を持つリンクlを選択
  - lを追加し、Aを更新

全部のdestinationが
ソースに連結された



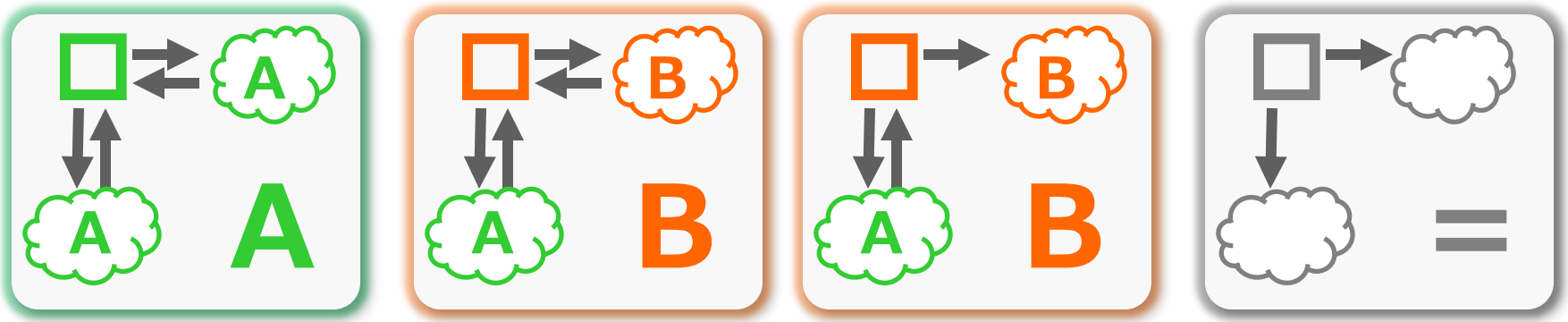
ボトルネック最大化のアルゴリズム (2)

- 各ソースについて、接続されたdestinationをパイプラインで辿れるまで追加を続ける
  - 1つのdestinationに複数のソースが接続された場合は、スループットが大きい方を選択する
- 複数のソースを用いて効率的にマルチキャストできるパイプライン群が求まった
  - 多くの場合、合計スループットも最大となる



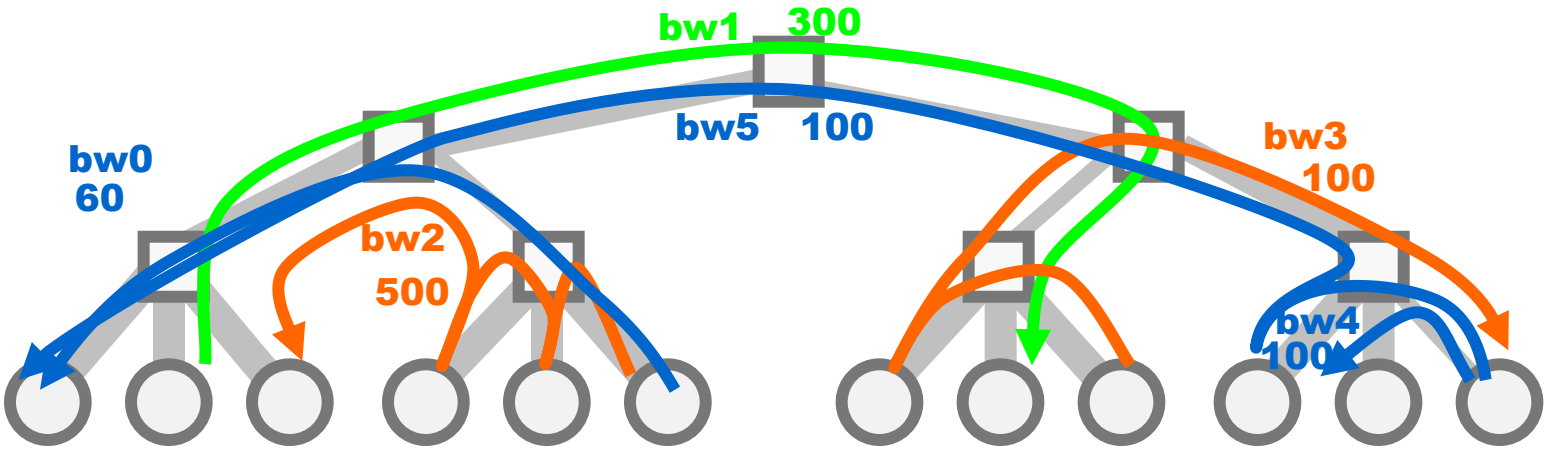
パイプラインが構成可能な条件

- 以下の2条件を考える
  - A: 始点からdestinationを一直線で辿って戻れる
  - B: 始点からdestinationを一直線で辿れる
- 全てのsourceがBを満たせばよい



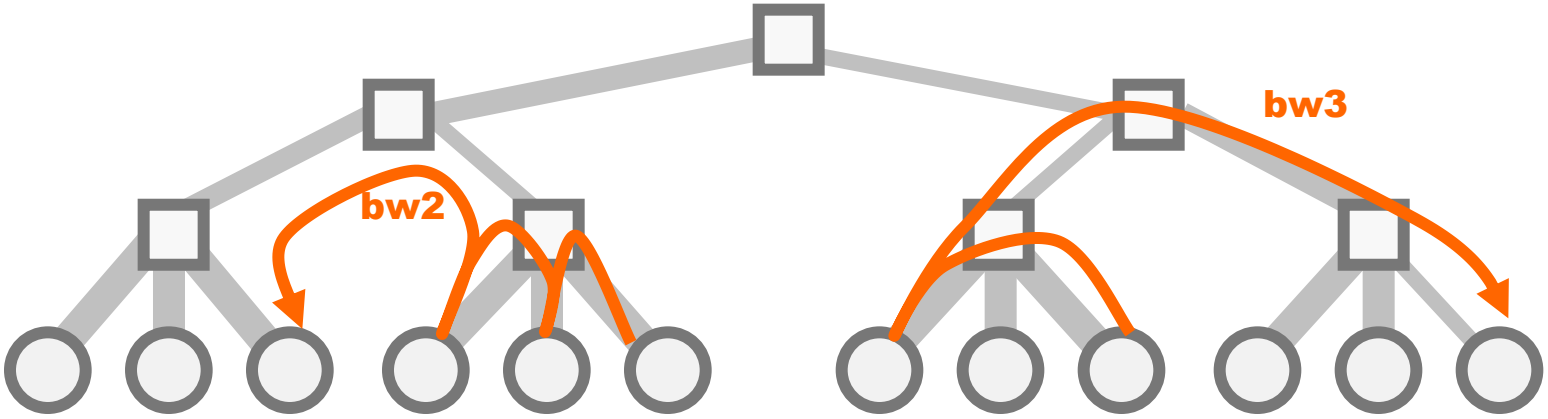
提案するアルゴリズム

1. 全destinationとsourceを結ぶ経路を作る
   - 1転送ずつ独立に最小スループット最大化問題を解いて、パイプラインを構築する
2. 線形計画法で合計スループットを最大化する
3. 得られた計画に対し、近傍探索で改善を行う



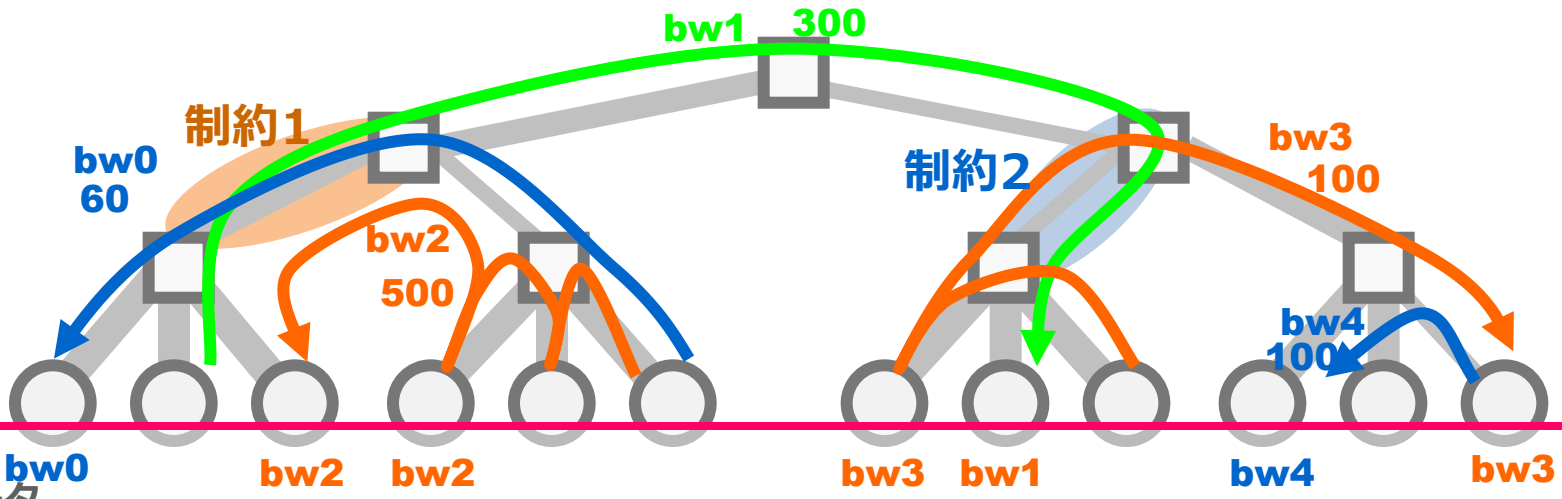
複数データの転送計画

- １つのデータ転送ずつ、独立に「最小バンド幅最大化」問題を解いて、転送計画を立てる
  - ◆複数のソースを利用してパイプライン群を構成



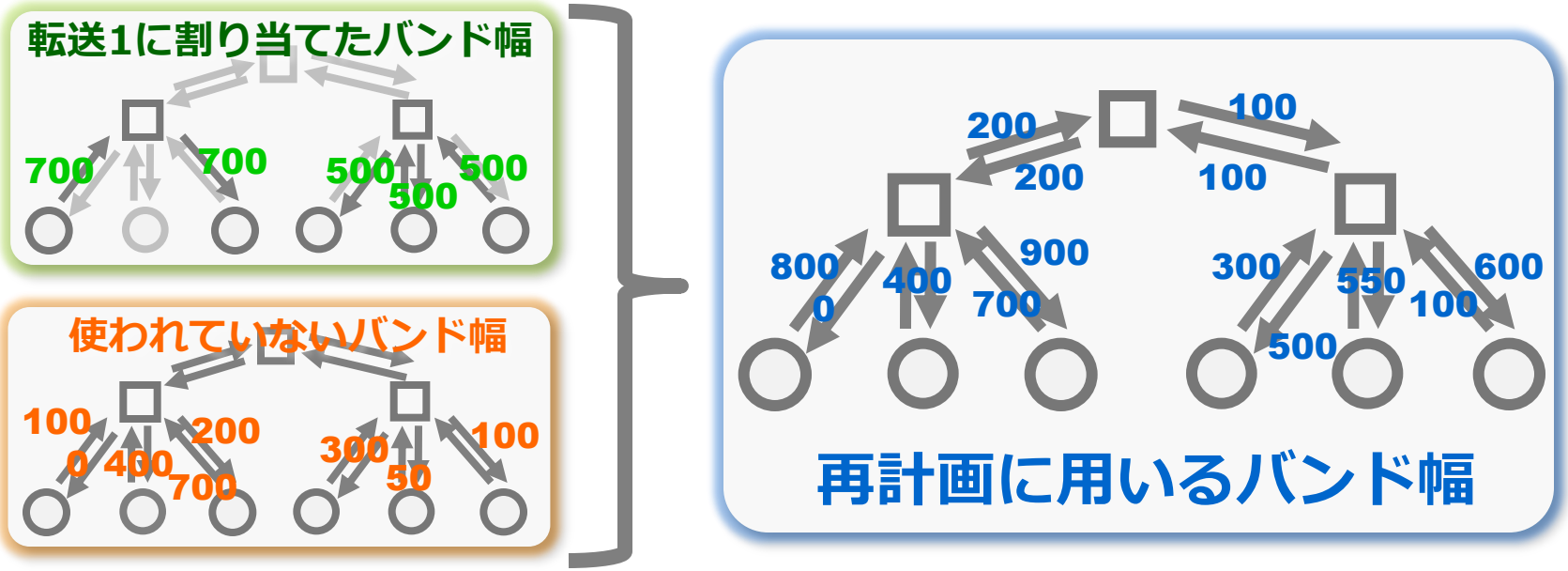
線形計画法でスループット最大化

- 線形計画法を用いて合計スループットを最大化するよう、各転送にバンド幅を割り当てる
  - 目的関数: (bw0 + bw1 + bw2 * 3 + bw3 * 2 + bw4)
  - 制約1: bw0 + bw1 ≤ (const)
  - 制約2: bw1 + bw3 ≤ (const)
  - …



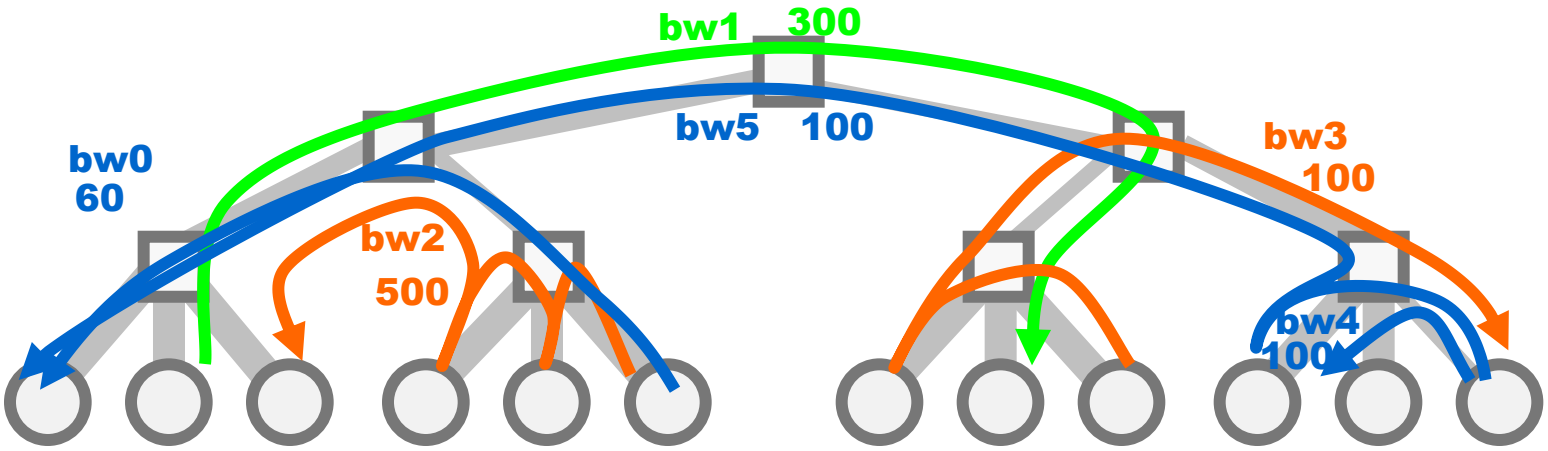
転送の再計画（1）

- 1つずつ転送を再計画
  - ある転送に割り当てられたバンド幅と、使われていないバンド幅を加えたバンド幅マップTを作成



転送の再計画 (2)

- Tを元に最小バンド幅最大問題を解く
- 得られた計画に対し再び線形計画法を解く
- 合計スループットが改善された場合のみ、変更を採用する　(山登り法)



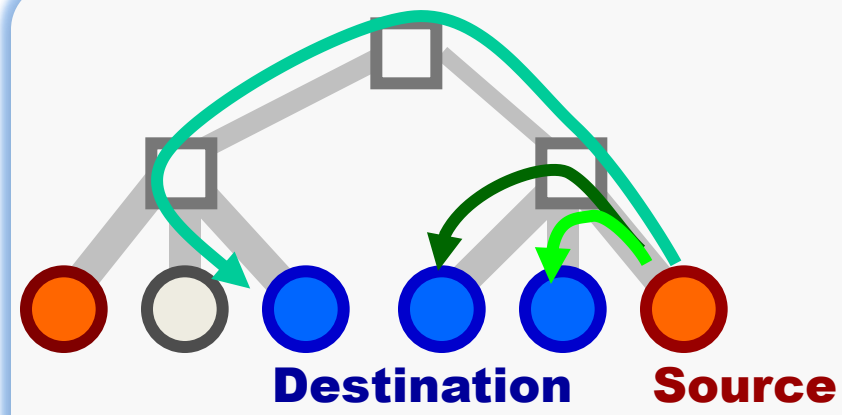
発表の流れ

- はじめに
- 問題設定
- アルゴリズム
- 実験・考察
- 終わりに

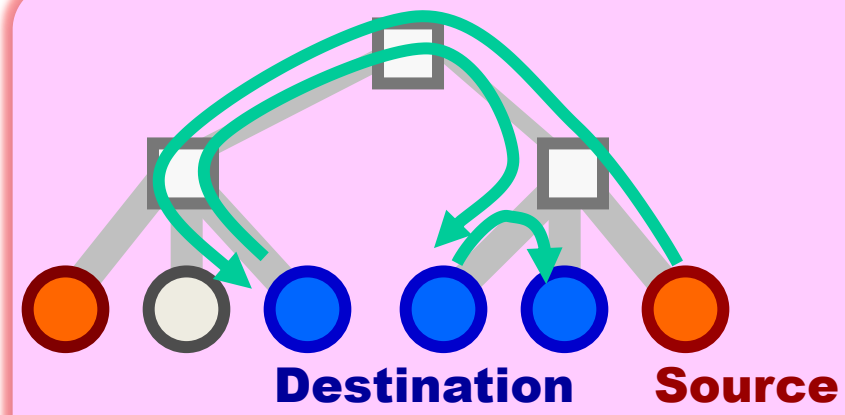
実験・評価

- ランダムに生成したバンド幅マップ・転送要件(srcs, dests)に基づいて、以下の4つのアルゴリズムを比較するシミュレーションを行った
  1. 各destinationがランダムにsourceを選択し、直接データを送信 (Random-flat tree)
  2. 各destinationは最も大きなバンド幅で接続できるsourceを選択し、同じsourceを用いるノード間でランダムなパイプラインを構築 (Random-pipeline)
  3. 各destinationは最も大きなバンド幅で接続できるsourceを選択し、トポロジに沿ってパイプラインを構築 (Topology-aware pipeline)
  4. 3の後、パイプラインの再計画を実行 (Improved pipeline, 提案手法)

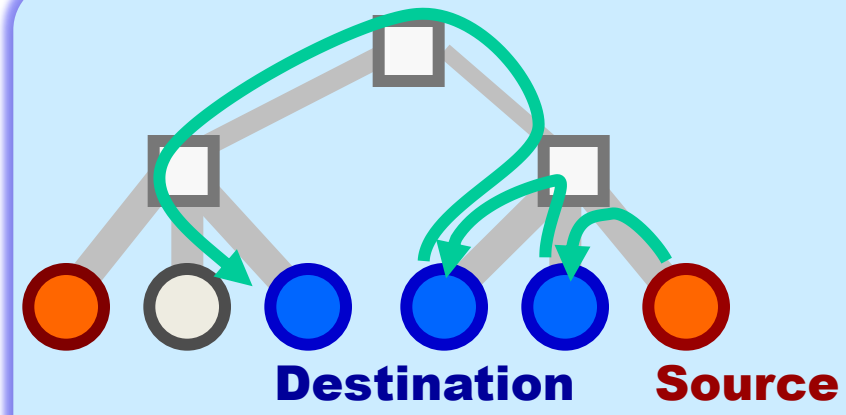
各手法のイメージ



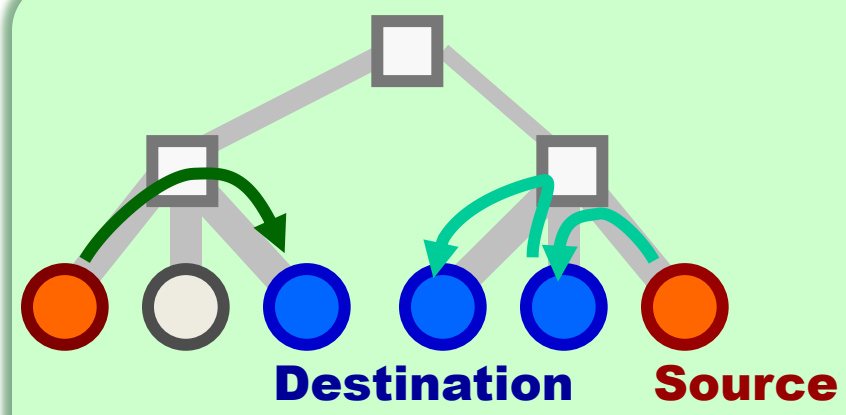
Random-flat Tree



Random-pipeline



Topology-aware Pipeline



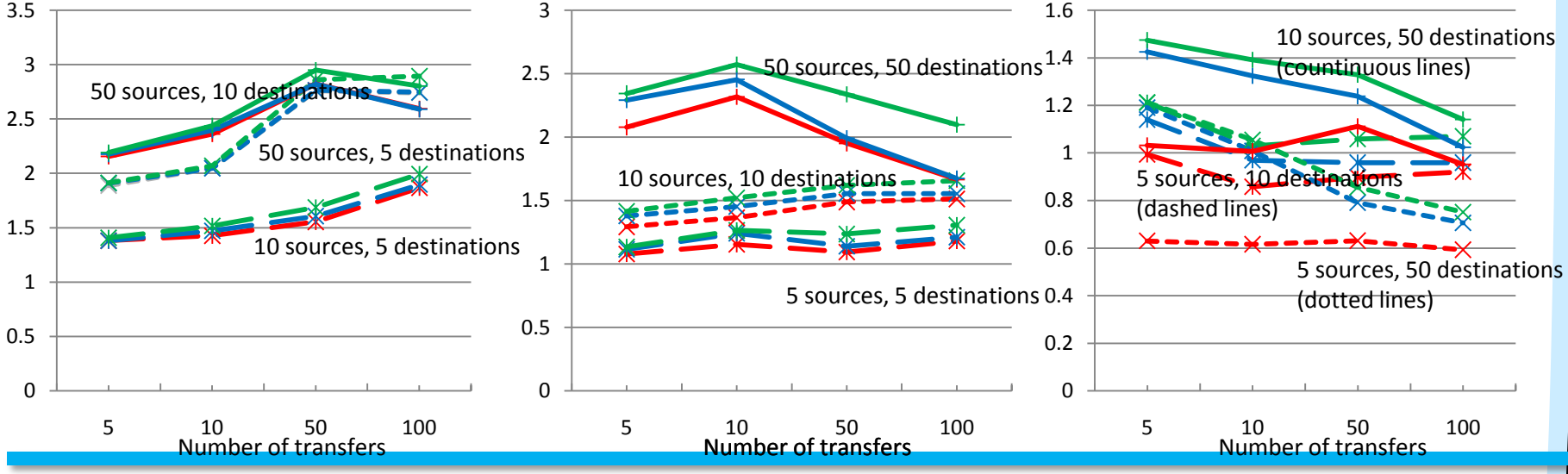
Improved
(他の転送を考慮して再計画)

実験結果(1)

- 36の条件で実験で10回ずつ実験した
  - Random-flatとのスループットの比率をプロット

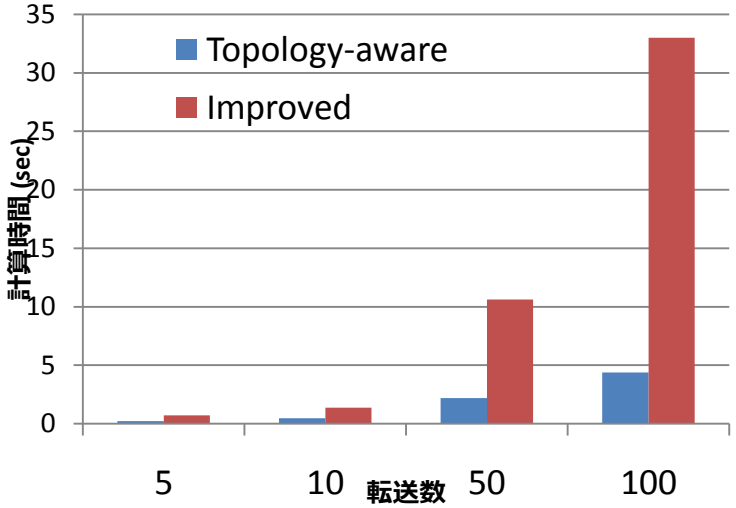
(■ Random pipeline ■ Topology-aware pipeline ■ Improved pipeline)

  - ランダムな手法は1問題につき10回の平均を取った
- 提案手法が最も高い性能を示した
  - Random-flatに比べ最大2.9倍，平均1.7倍
  - Random-pipelineに比べ最大1.7倍，平均1.3倍
  - Topology-aware に最大1.3倍、平均1.05倍



実験結果(2)

- 予測したほどの性能向上は図れなかった
  - 目的関数をもっと意識した、既存の最適化手法を活用したい
- 計算時間は50転送で10秒、100転送で32秒
  - 50転送では十分な実行速度
  - 転送のを局所性によってクラスタリングするなどの手法も考えたい



おわりに

- トポロジ情報を利用し、スループットを向上させるような転送計画アルゴリズムを提案した
  - 複数のデータ・複数のソース
  - 複数ソースを活用し、ボトルネックバンド幅を最大化
  - 複数転送間でのリンクの競合を検知し、回避する
- シミュレーションによって評価を行った
  - トポロジの考慮によって、ランダムな手法に比べ平均1.7倍、パイプラインを用いるがトポロジを考慮しない方法に比べ1.3倍のスループット改善が図られた
  - 実行時間は50転送で10秒であった

今後の計画

- 本転送計画スケジュラーを利用した、効率的なタスクスケジュラーを計画したい
  - 本スケジューリングアルゴリズムにより、与えられた転送群に対し、実行時間の予測が可能になる
  - タスク配置によって必要な転送群は変化する
  - 様々なタスク配置のうち、最も効率よくデータ転送を行える配置を求めるスケジュラーを計画する