都鳥: メモリキャッシュによるライブマイグレーションの高速化

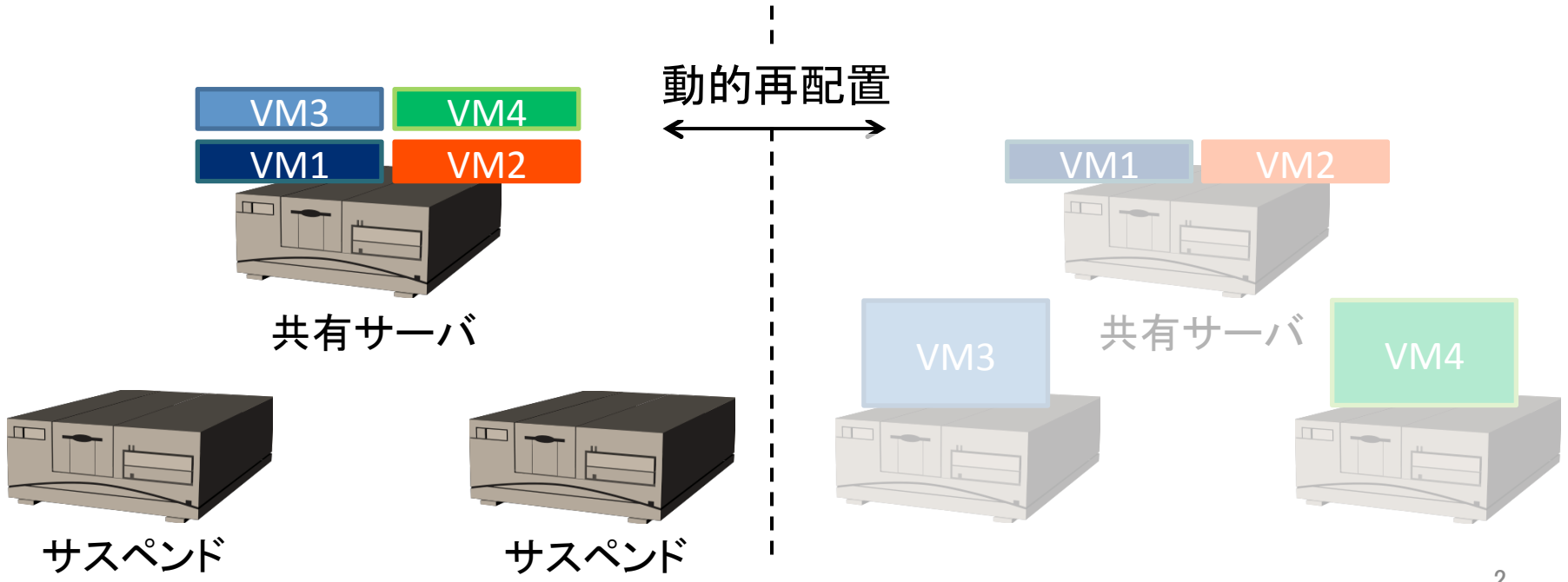
第30回仮想化実装技術勉強会

2011年12月19日

穐山空道 (東大 情報理工 創造情報 M2)

背景: VMの動的再配置による クラウド効率化(1/2)

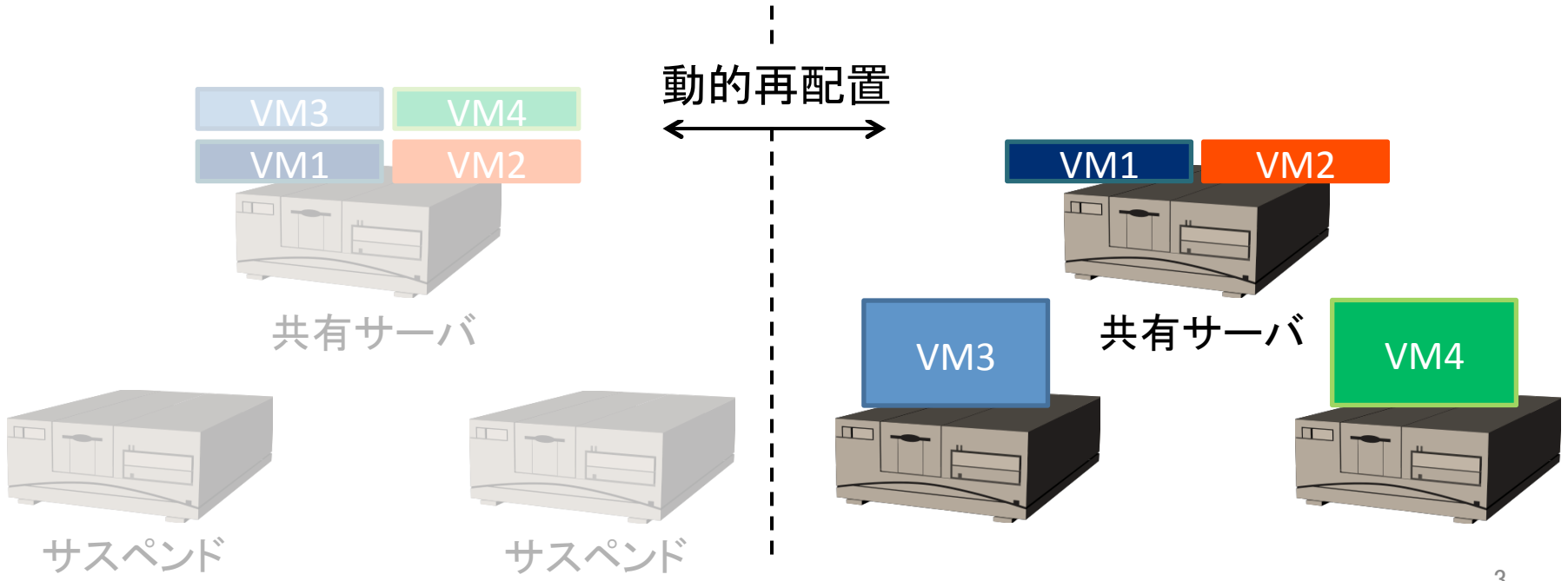
- Dynamic VM Consolidation[1]
  - 低負荷 →共有サーバに集約し電力削減
  - 高負荷 →分散配置し性能保障



[1] Reactive consolidation of virtual machines enabled by postcopy live migration, *VTDC2011,* Hirofuchi *et al.*

背景: VMの動的再配置による クラウド効率化(1/2)

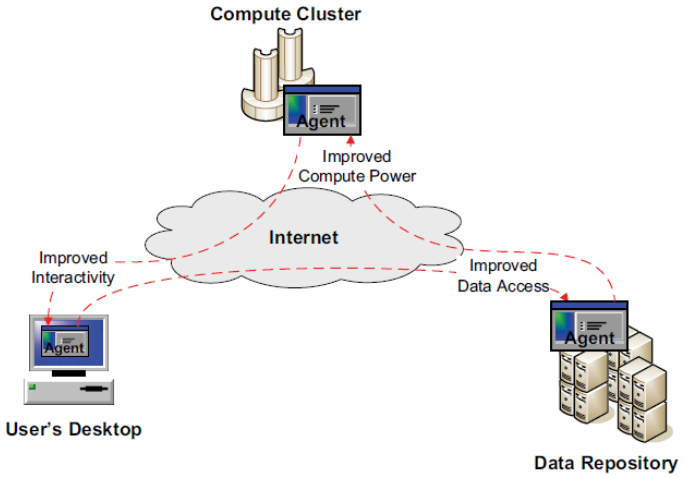
- Dynamic VM Consolidation[1]
  - 低負荷 →共有サーバに集約し電力削減
  - 高負荷 →分散配置し性能保障



[1] Reactive consolidation of virtual machines enabled by postcopy live migration, *VTDC2011,* Hirofuchi *et al.*

背景: VMの動的再配置による クラウド効率化(2/2)

- Memory Buddies[2]
  - メモリの似たVMを集約しメモリ共有→メモリ使用量削減

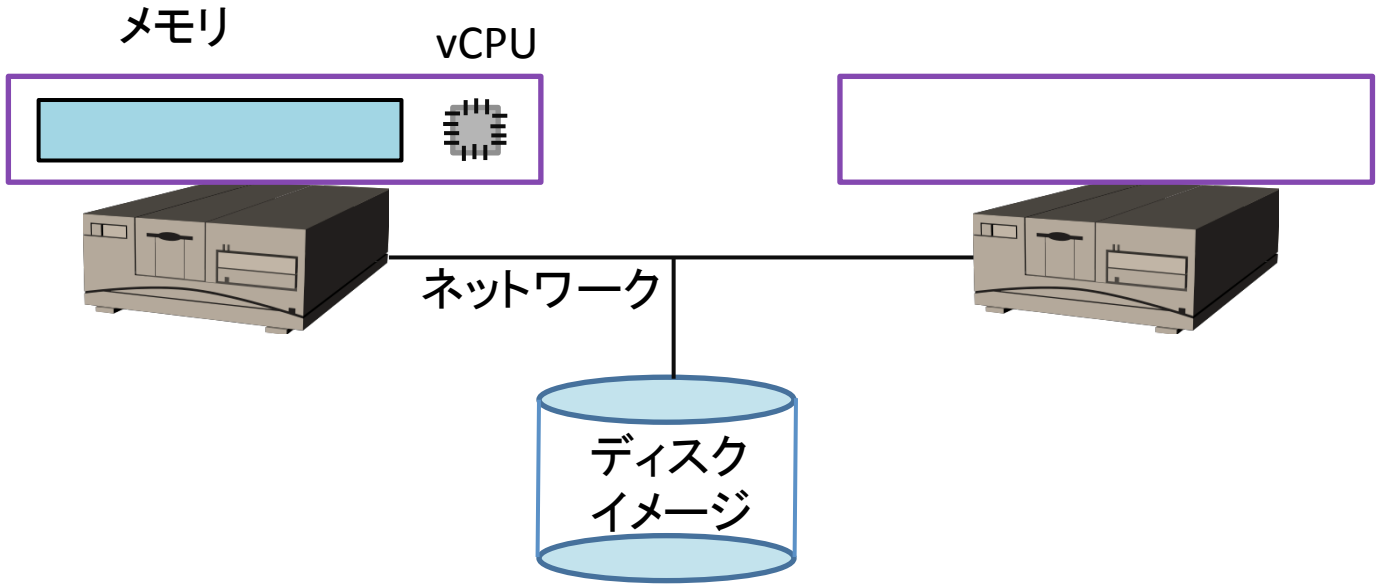


- SnowBird[3]
  - VMがデスクトップ、計算ノード、データベース間を移動→アプリケーション実行時間削減

[2] Interactive Resource-Intensive Application Made Easy, *Middleware2007,* Largar-Cavilla *et al.*
[3] Memory Buddies: Exploiting Page Sharing for Smart Collocation in Virtualized Data Centers, *VEE2009,* Wood *et al.*

Pre-copy Live Migration [4]

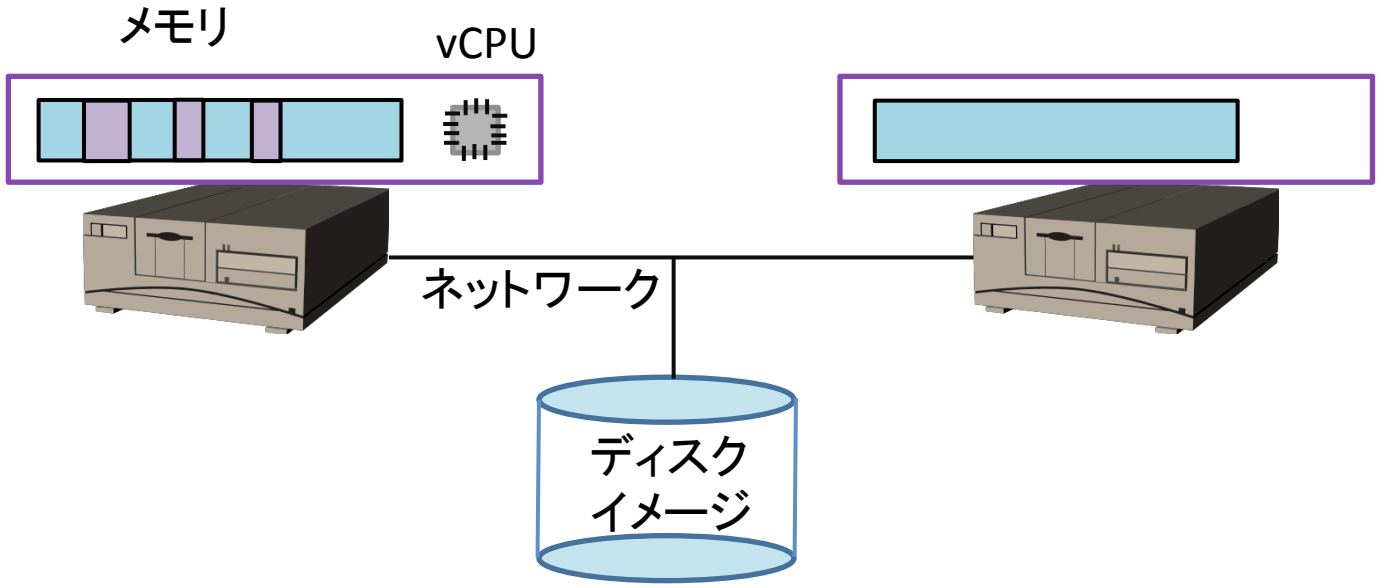
- 全メモリをコピーした後、実行ホストを移動
  - コピー中に更新されたメモリ → 再コピー
  - コピー量が十分減少 → VMを停止し残りをコピー



[4] Live Migration of Virtual Machines, *NSDI2005,* Clark *et al.*

Pre-copy Live Migration [4]

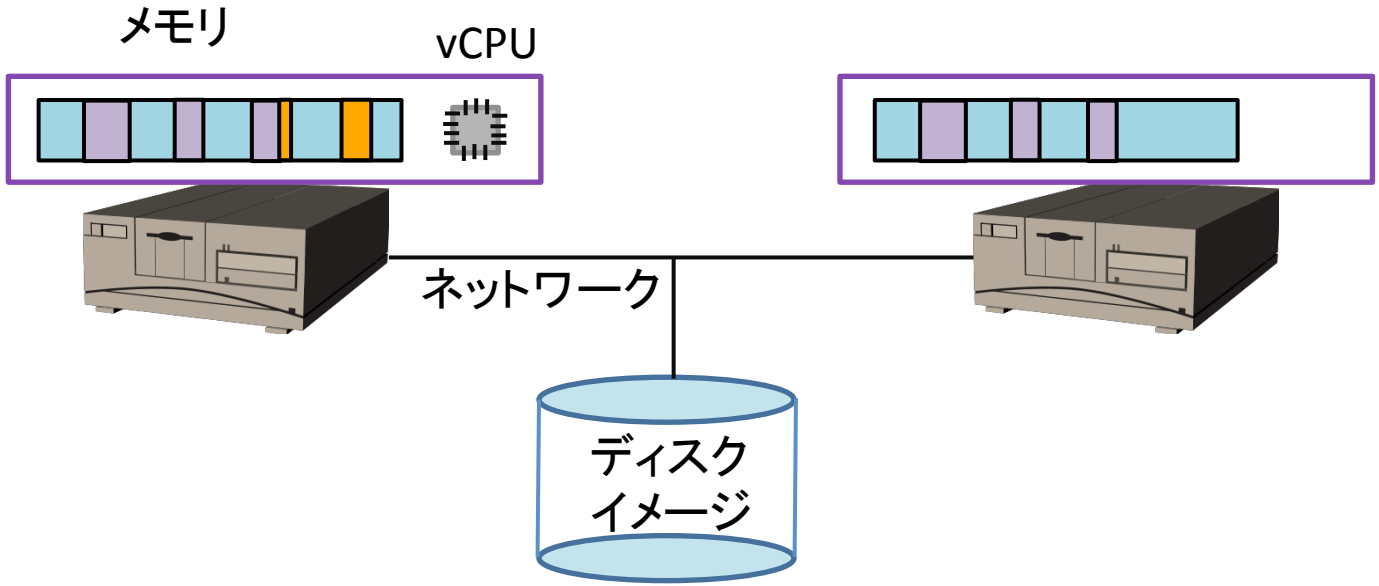
- 全メモリをコピーした後、実行ホストを移動
  - コピー中に更新されたメモリ → 再コピー
  - コピー量が十分減少 → VMを停止し残りをコピー



[4] Live Migration of Virtual Machines, *NSDI2005,* Clark *et al.*

Pre-copy Live Migration [4]

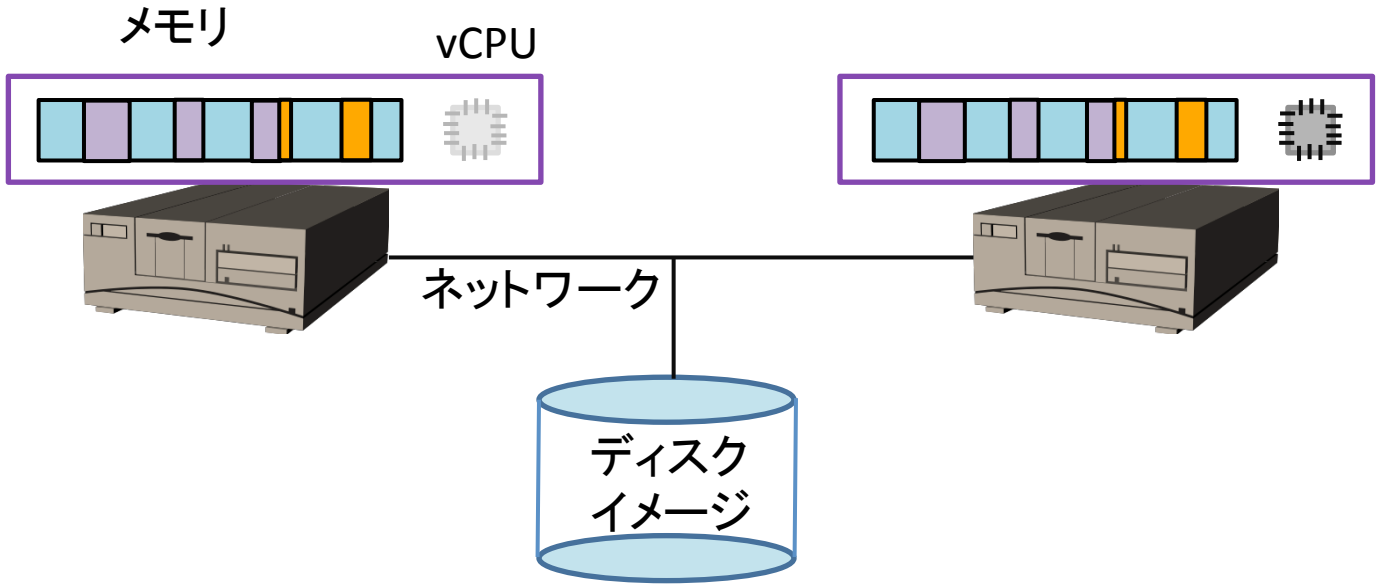
- 全メモリをコピーした後、実行ホストを移動
  - コピー中に更新されたメモリ → 再コピー
  - コピー量が十分減少 → VMを停止し残りをコピー



[4] Live Migration of Virtual Machines, *NSDI2005,* Clark *et al.*

Pre-copy Live Migration [4]

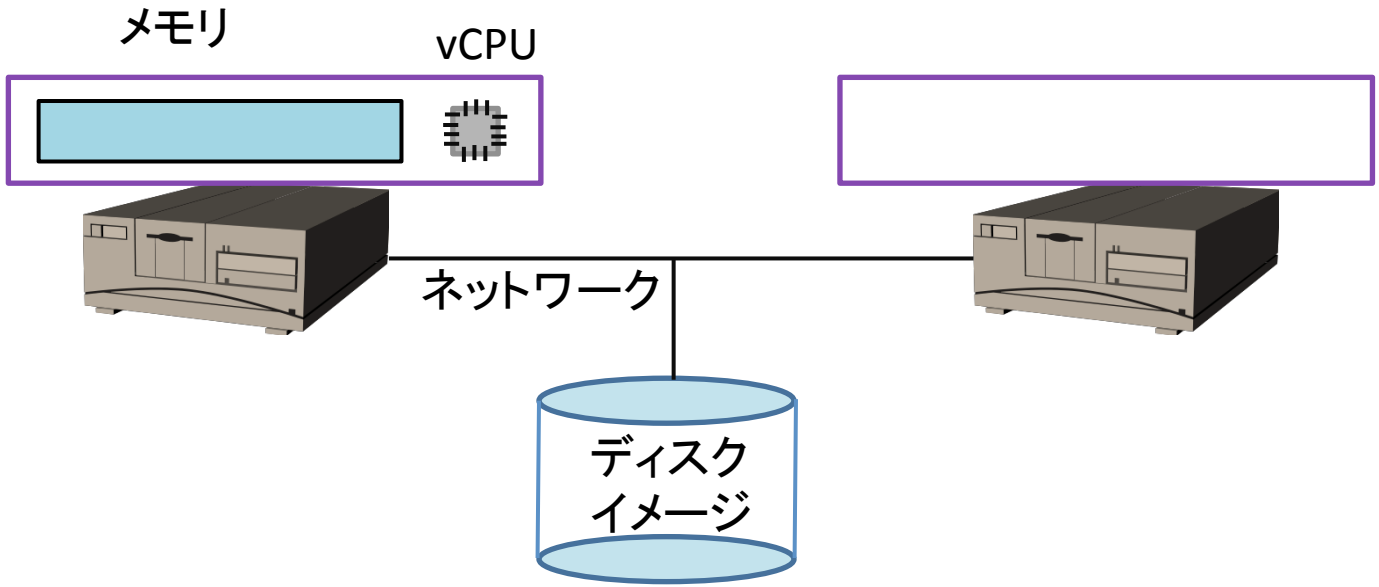
- 全メモリをコピーした後、実行ホストを移動
  - コピー中に更新されたメモリ → 再コピー
  - コピー量が十分減少 → VMを停止し残りをコピー



[4] Live Migration of Virtual Machines, *NSDI2005,* Clark *et al.*

Post-copy Live Migration [5][6]

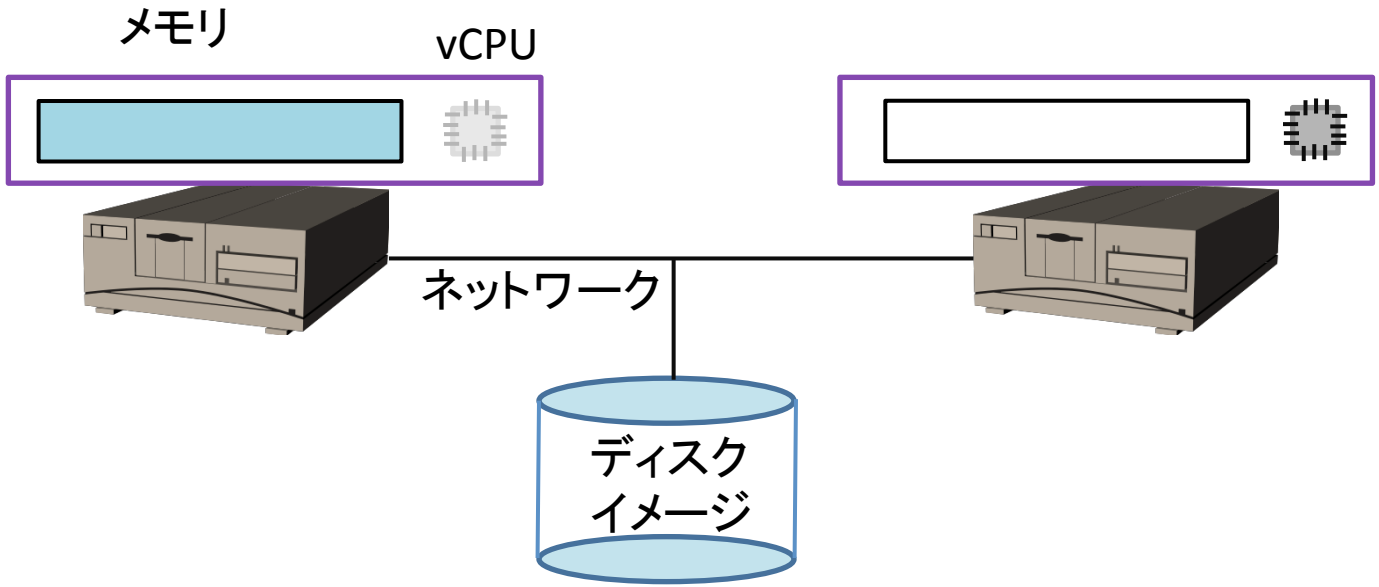
- 実行ホストを移動した後、全メモリをコピー
  - ページフォールトによるon-demandなコピー
  - バックグラウンドでのコピー



[5] SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *EuroSys2008,* Largar-Cavilla *et al.*
[6] Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *CCGrid2010,* Hirofuchi *et al.*

Post-copy Live Migration [5][6]

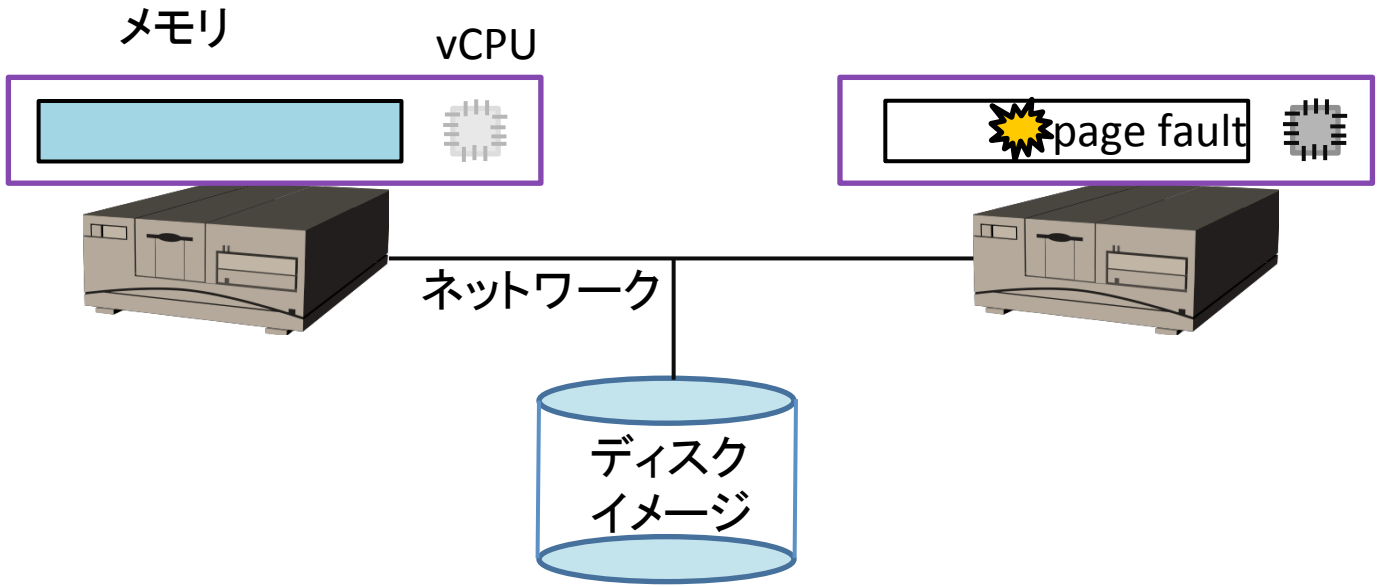
- 実行ホストを移動した後、全メモリをコピー
  - ページフォールトによるon-demandなコピー
  - バックグラウンドでのコピー



[5] SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *EuroSys2008,* Largar-Cavilla *et al.*
[6] Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *CCGrid2010,* Hirofuchi *et al.*

Post-copy Live Migration [5][6]

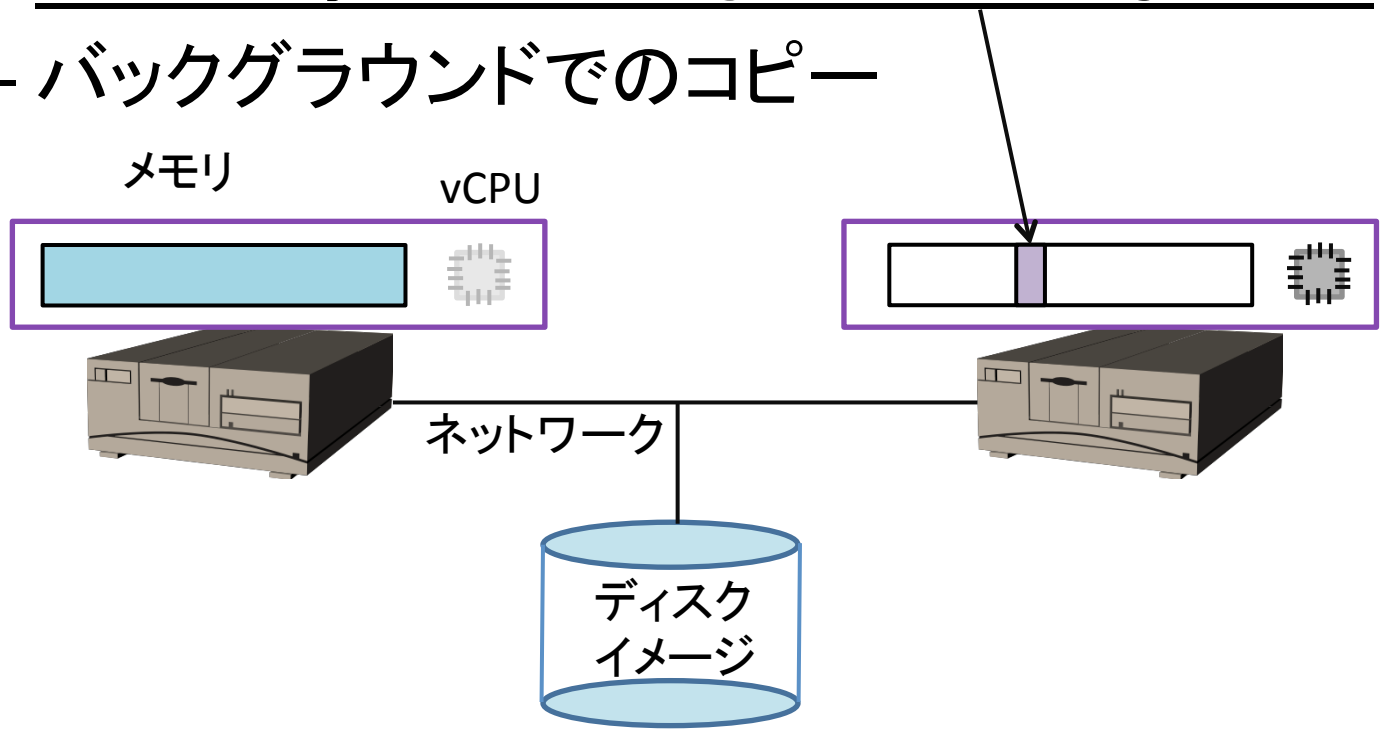
- 実行ホストを移動した後、全メモリをコピー
  - ページフォールトによるon-demandなコピー
  - バックグラウンドでのコピー



[5] SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *EuroSys2008,* Largar-Cavilla *et al.*
[6] Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *CCGrid2010,* Hirofuchi *et al.*

Post-copy Live Migration [5][6]

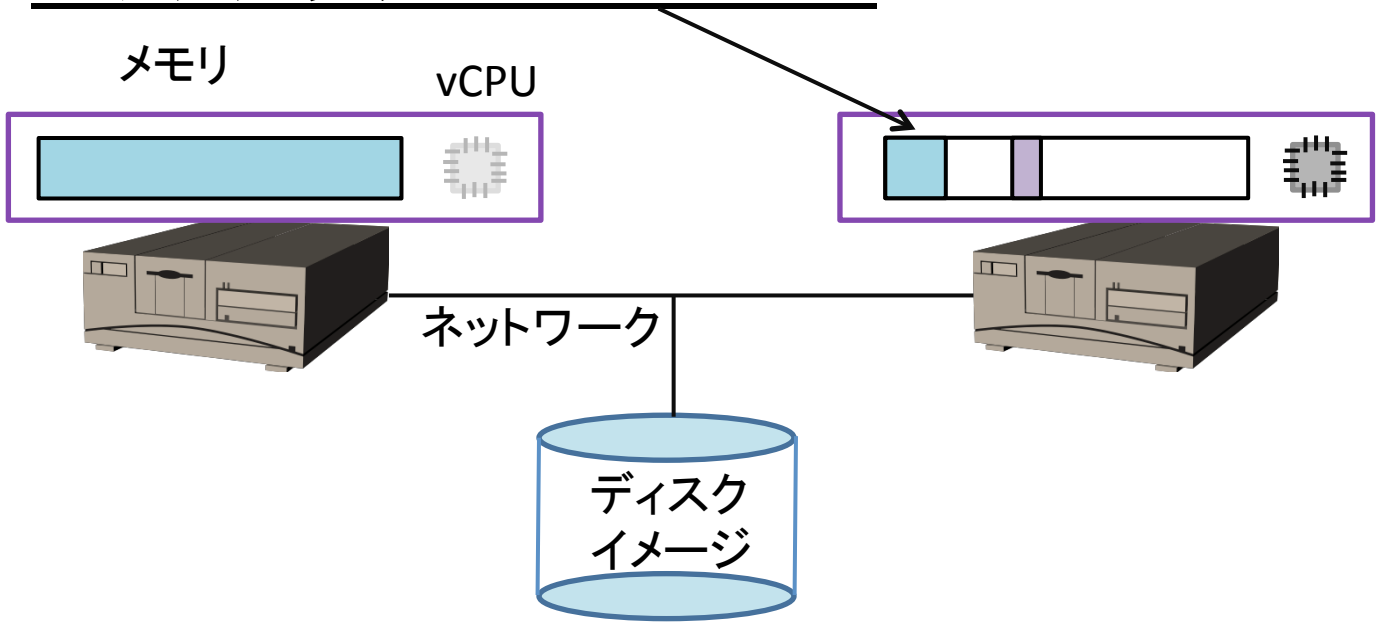
- 実行ホストを移動した後、全メモリをコピー
  - ページフォールトによるon-demandなコピー
  - バックグラウンドでのコピー



[5] SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *EuroSys2008,* Largar-Cavilla *et al.*
[6] Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *CCGrid2010,* Hirofuchi *et al.*

Post-copy Live Migration [5][6]

- 実行ホストを移動した後、全メモリをコピー
  - ページフォールトによるon-demandなコピー
  - バックグラウンドでのコピー



[5] SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing, *EuroSys2008,* Largar-Cavilla *et al.*
[6] Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *CCGrid2010,* Hirofuchi *et al.*

Live Migrationの問題点

- 全メモリを転送する
  - VMのメモリサイズは小さくない（一般に数GB）
  - 長いマイグレーション時間（メモリサイズ/帯域幅）

1. マイグレーション中はアプリケーションパフォーマンス↓
   (workloadによって8%～30% [7])
2. サーバ集約時に複数のマイグレーション受け入れが同時に起こりやすくなる [8]

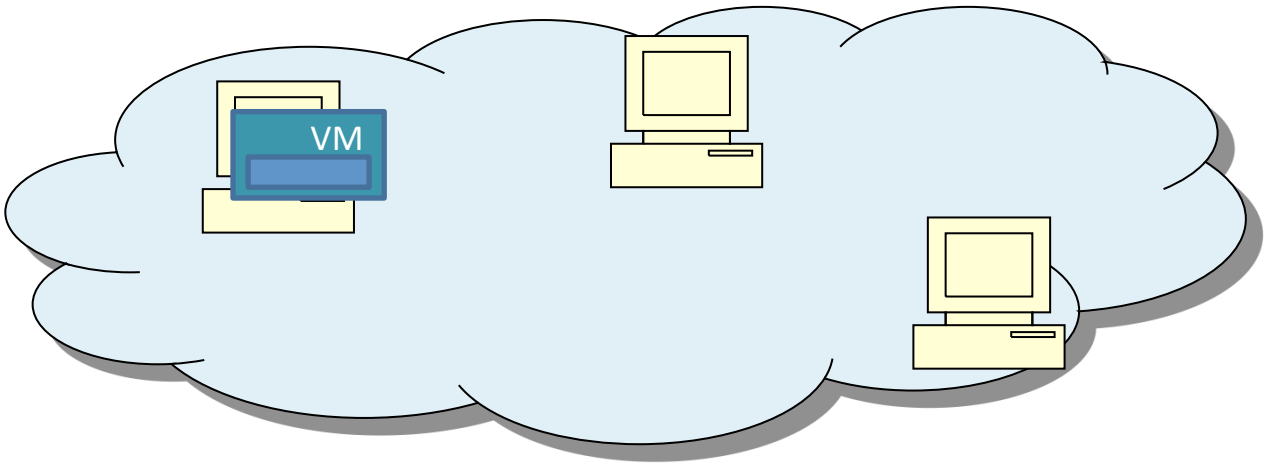
[7] Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments, *IEEE CLOUD 2011,* Ye *et al.*

[8] Performance Modeling of Concurrent Live Migration Operations in Cloud Computing Systems using PRISM Probabilistic Model Checker, *IEEE CLOUD 2011,* Kikuchi *et al.*

Memory Reusing: 着想

- VMの全メモリをコピーすることはコスト大（特に積極的な再配置環境下）
  - ポイント: VMが動的再配置されるとき、VMは一度実行されたホストに戻ってくる可能性

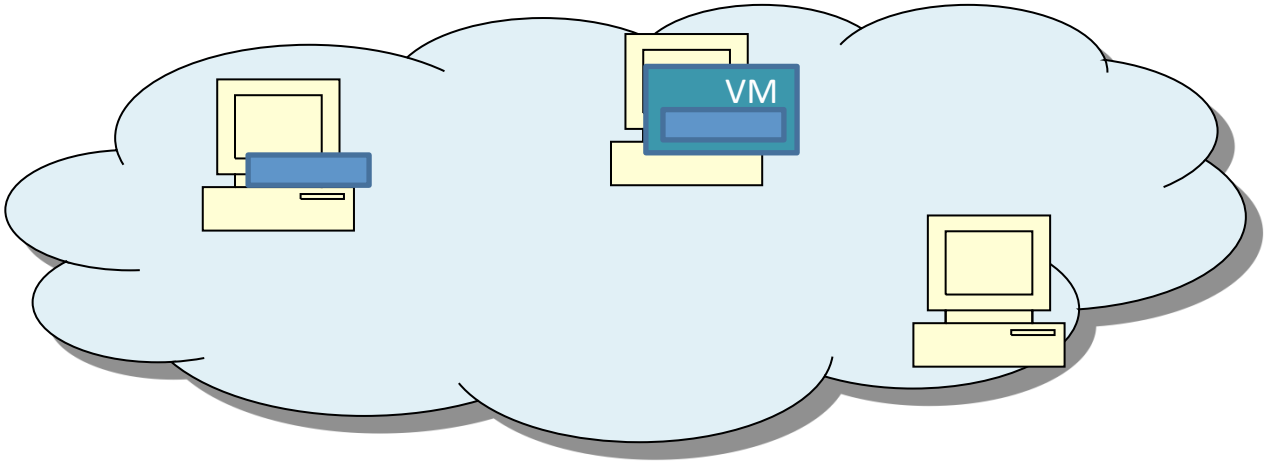
⇨ 更新されていないメモリページを再利用！



Memory Reusing: 着想

- VMの全メモリをコピーすることはコスト大（特に積極的な再配置環境下）
  - ポイント: VMが動的再配置されるとき、VMは一度実行されたホストに戻ってくる可能性

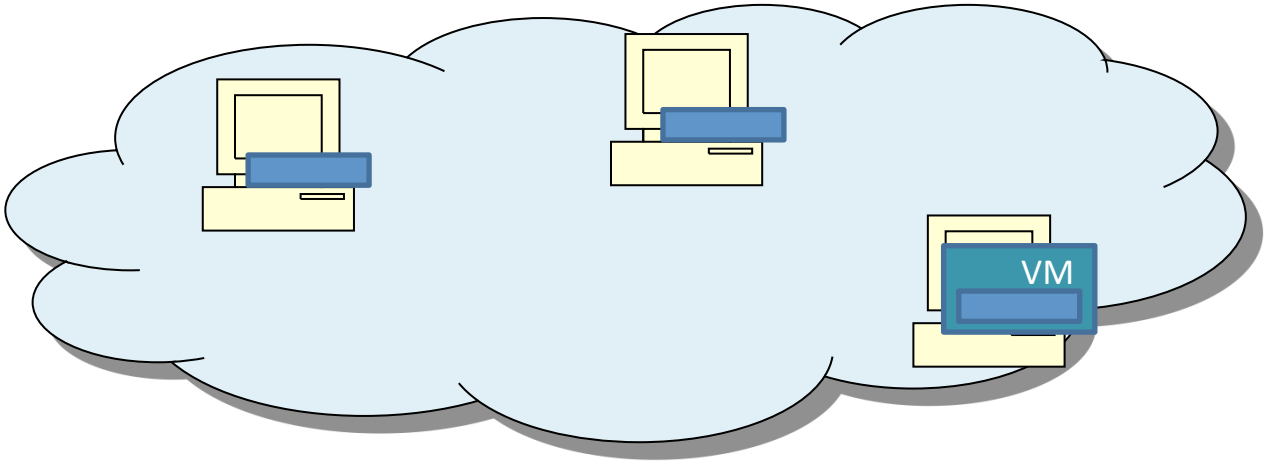
⇨ 更新されていないメモリページを再利用！



Memory Reusing: 着想

- VMの全メモリをコピーすることはコスト大（特に積極的な再配置環境下）
  - ポイント: VMが動的再配置されるとき、VMは一度実行されたホストに戻ってくる可能性

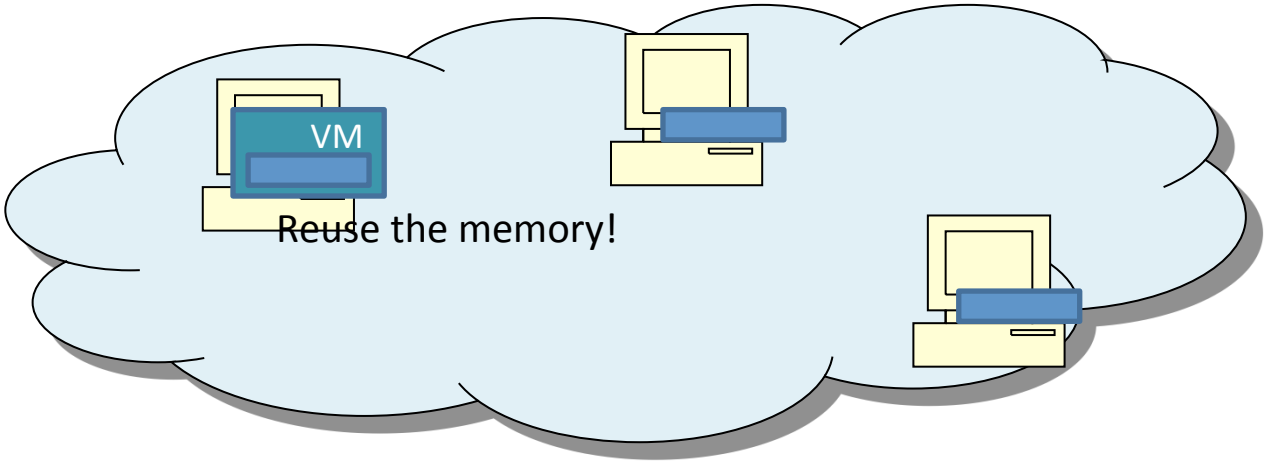
⇨ 更新されていないメモリページを再利用！



Memory Reusing: 着想

- VMの全メモリをコピーすることはコスト大（特に積極的な再配置環境下）
  - ポイント: VMが動的再配置されるとき、VMは一度実行されたホストに戻ってくる可能性

⇨ 更新されていないメモリページを再利用！



Memory Reusing: アルゴリズム

1. メモリページの世代
   - VMの起動時に0初期化
   - マイグレーション毎に更新ページの世代を+1
2. VMの移動時、移動元にメモリをキャッシュ
3. VMが戻ってくる時、キャッシュの世代と最新の世代を比較
   - 世代が同じページ → 再利用
   - 世代が違うページ → ネットワーク越しにコピー

*システム動作のスライドで図解

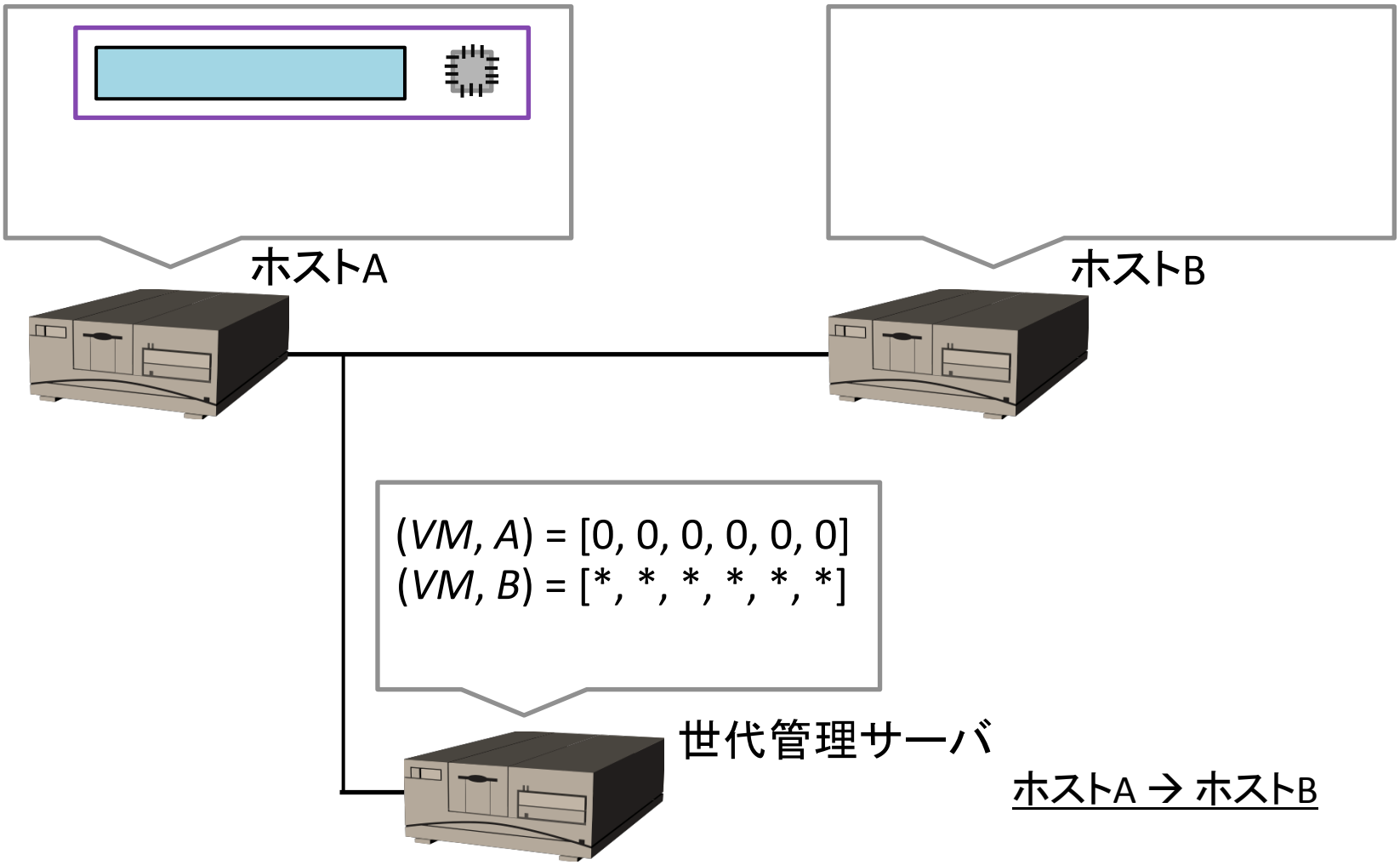
実装: 都鳥

- QEMU 0.13.0, KVMを利用
  - 変更前: 全メモリページを転送
  - 変更後: 更新されたページのみ転送(pre, post両方式に対応)
- 世代管理サーバを開発
  - 既存VMMへの変更を少なく
- 詳細
  - 世代は4バイト符号なし整数(実用上十分)
  - ホストはIPアドレスで区別
  - VMはuuidで区別

(*VM*,*host*)の組が、*host*にある*VM*のメモリキャッシュを表す

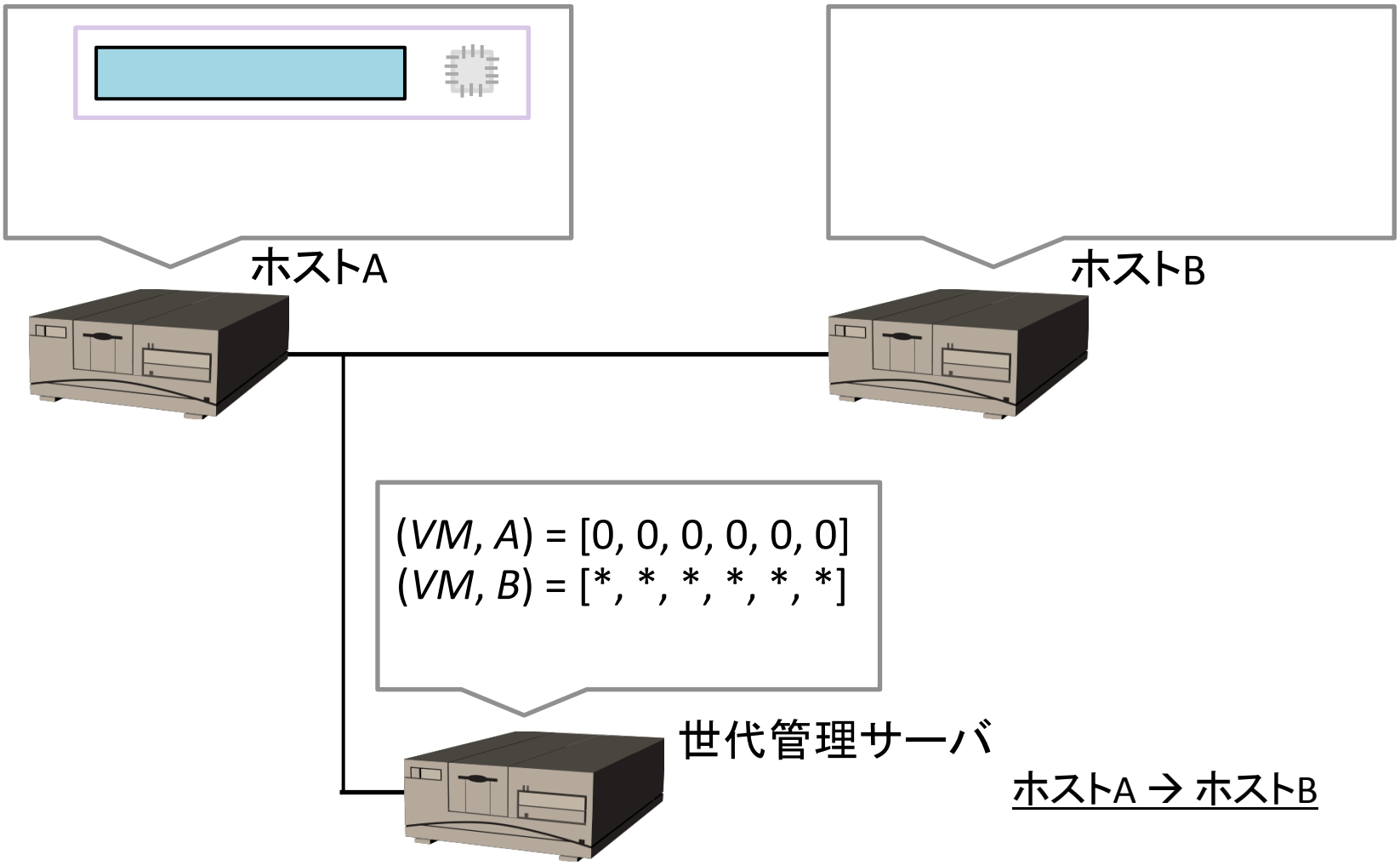
都鳥の動作: 最初のマイグレーション

0. VMが動作中

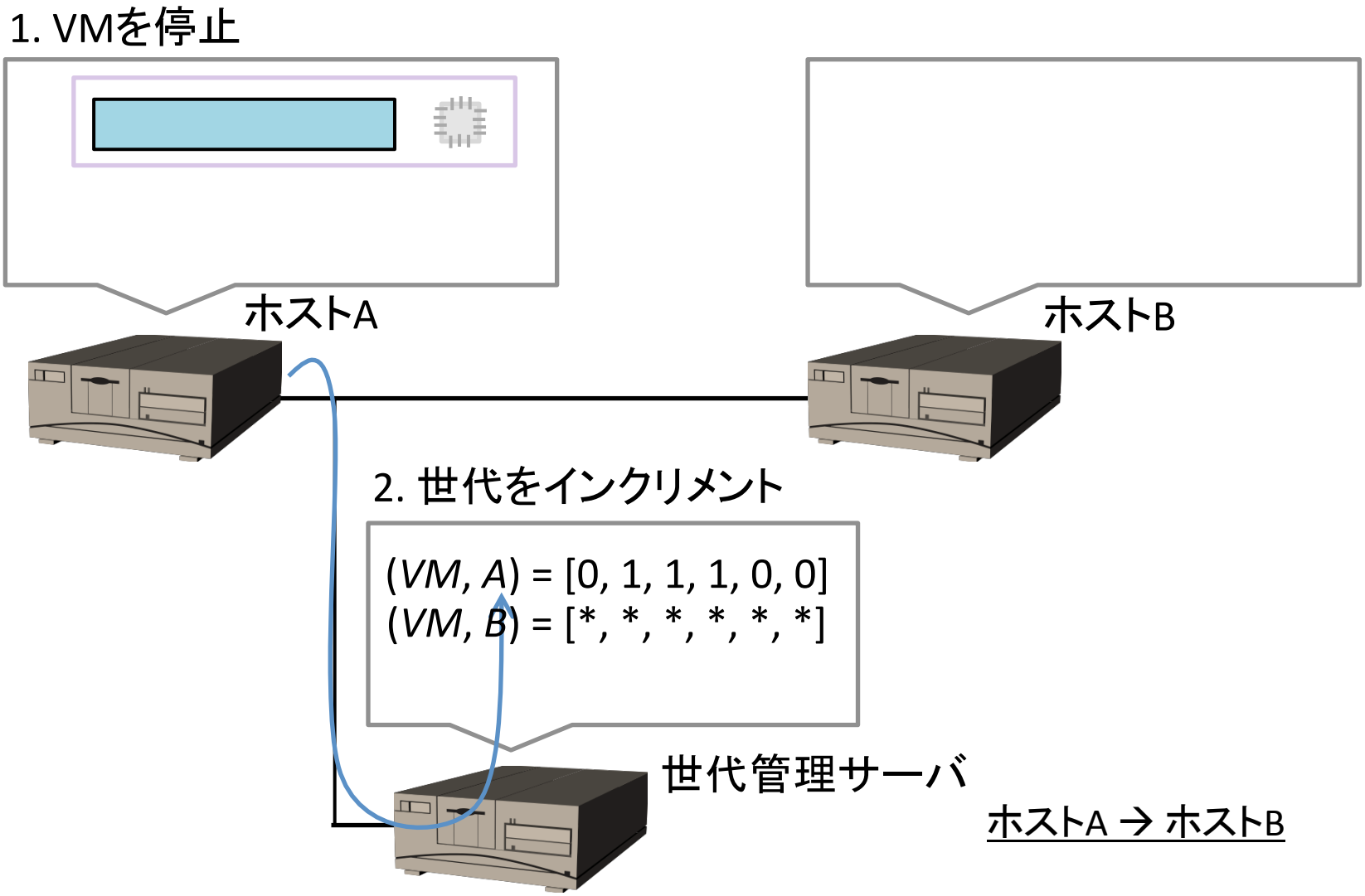


都鳥の動作: 最初のマイグレーション

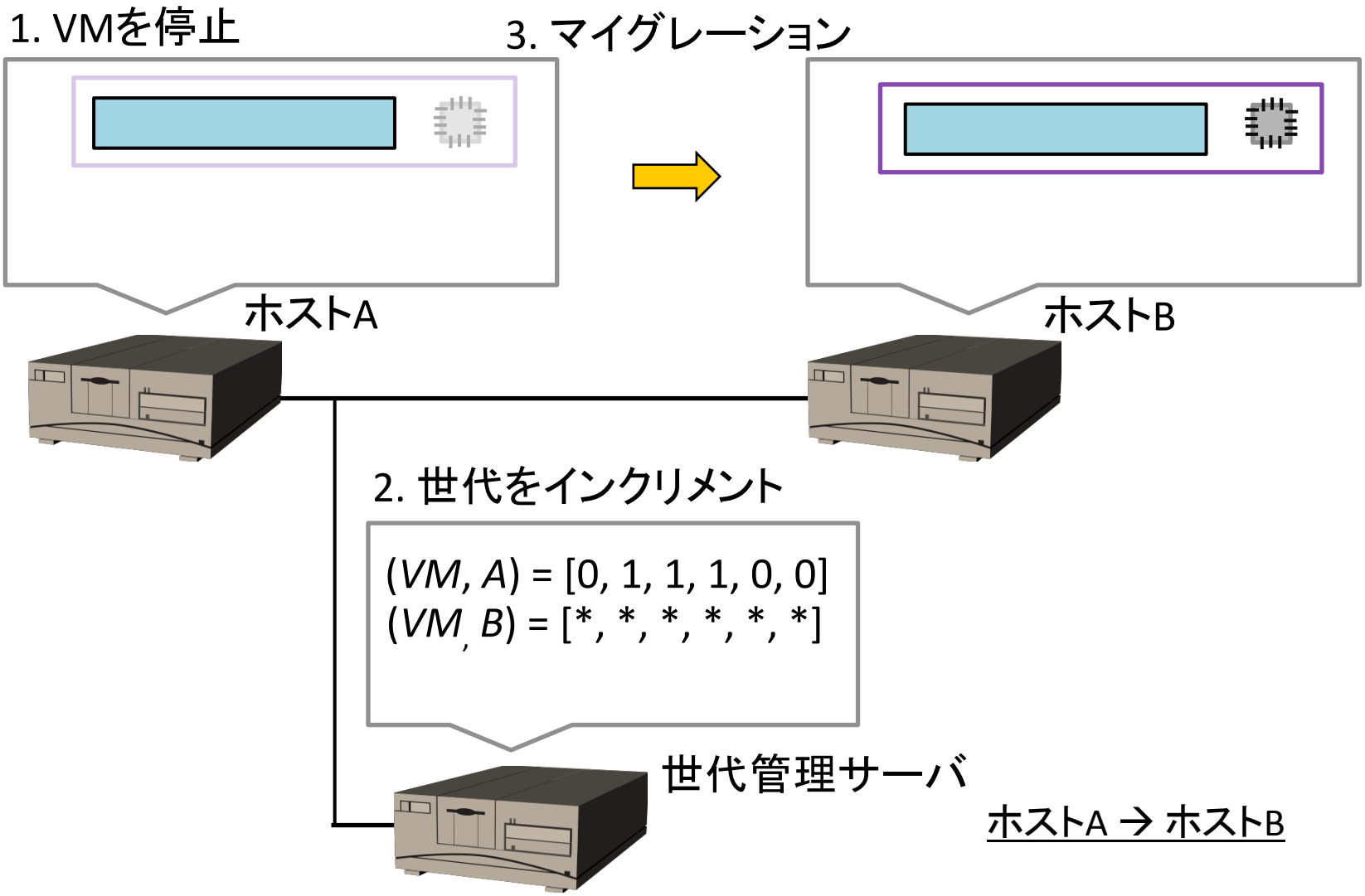
1. VMを停止



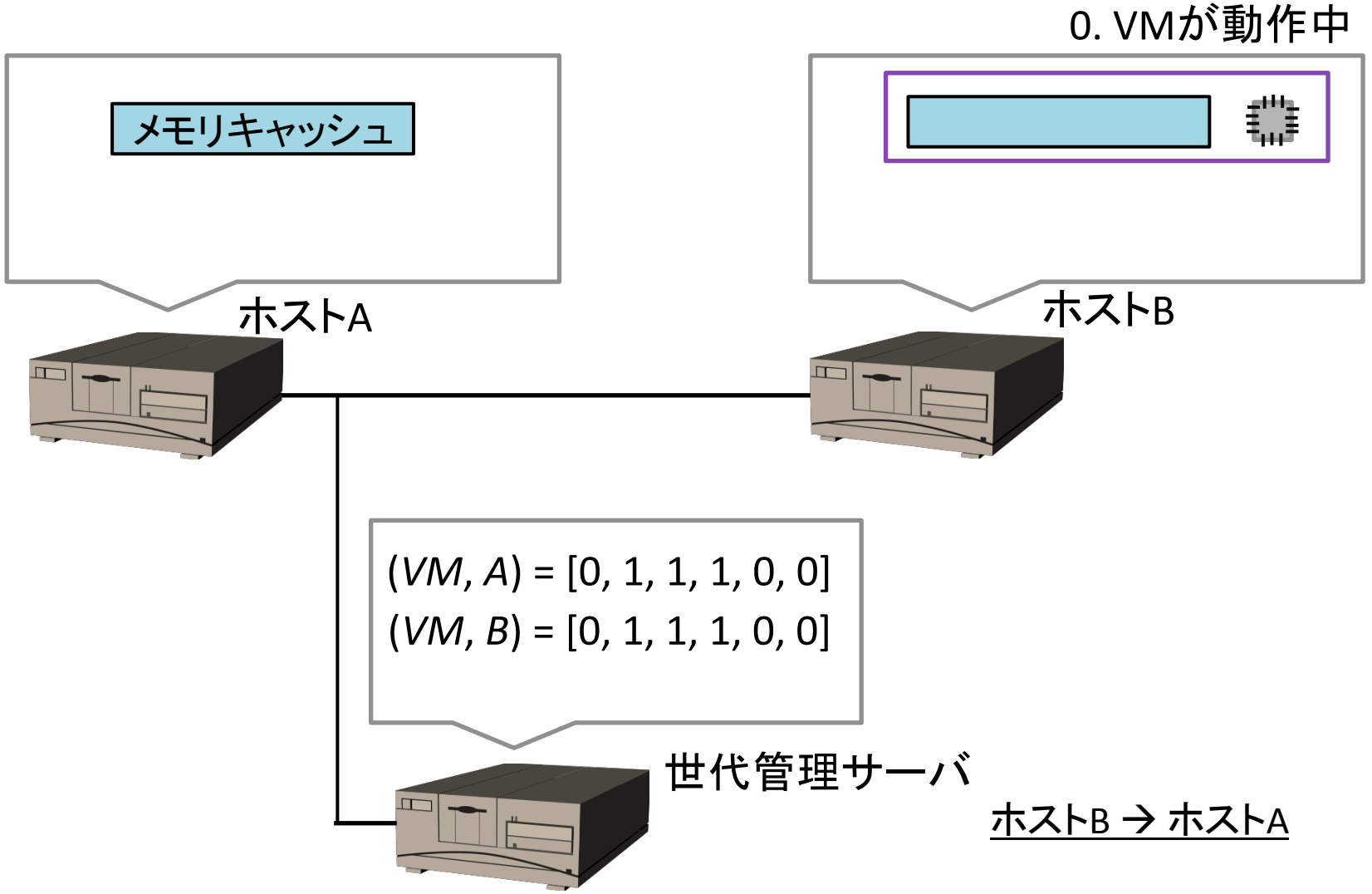
都鳥の動作: 最初のマイグレーション



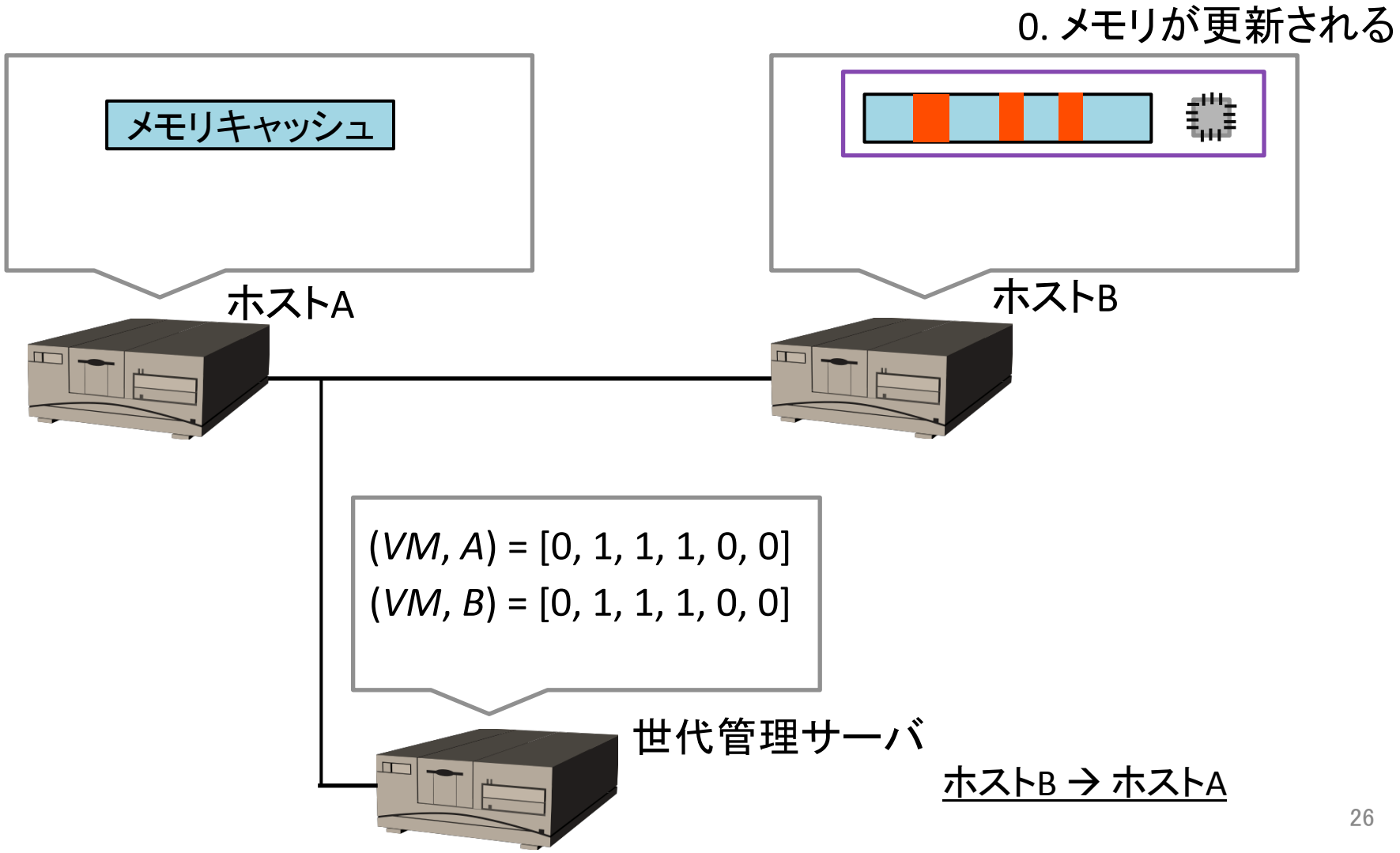
都鳥の動作: 最初のマイグレーション



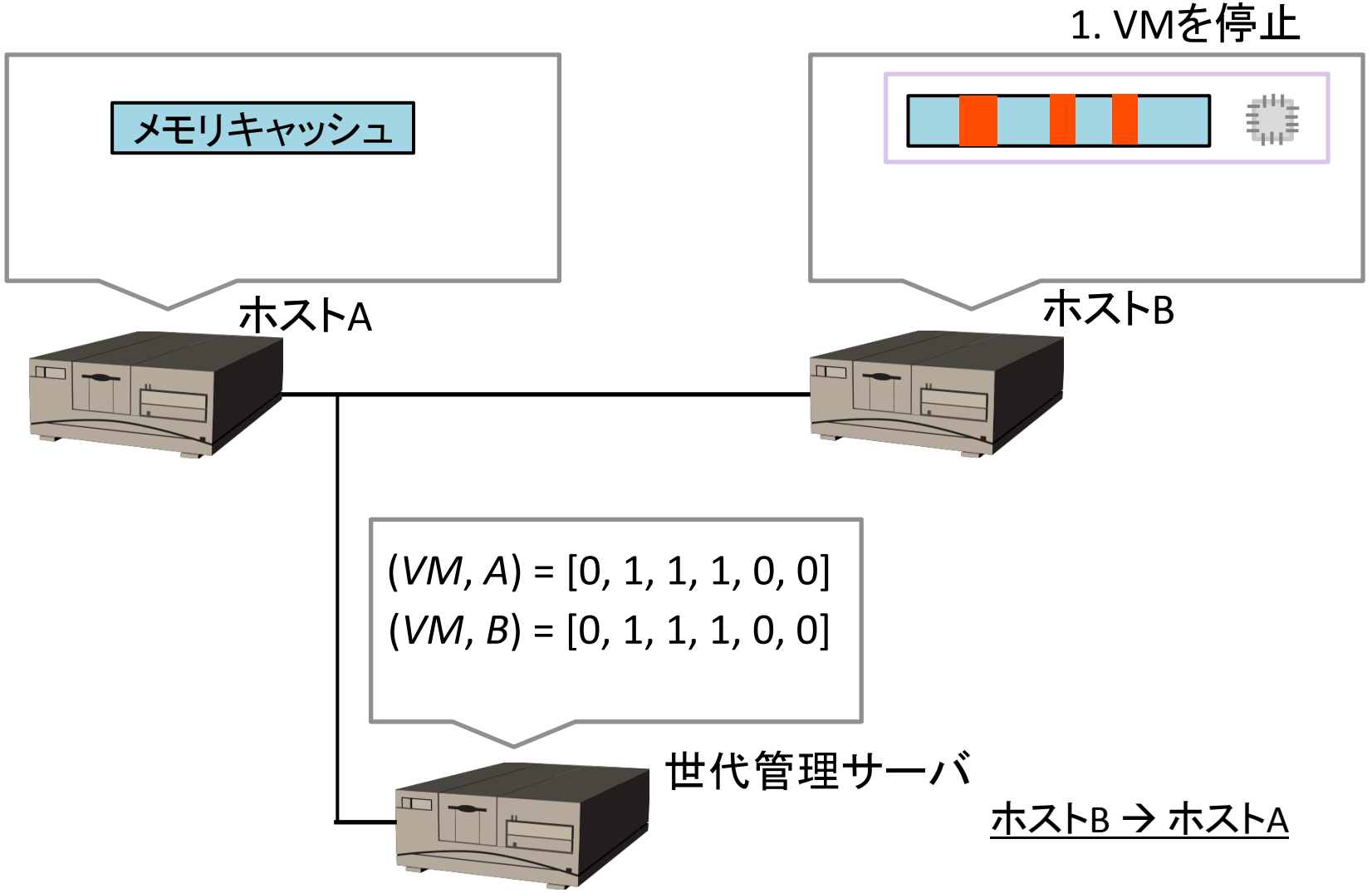
都鳥の動作: 戻るマイグレーション



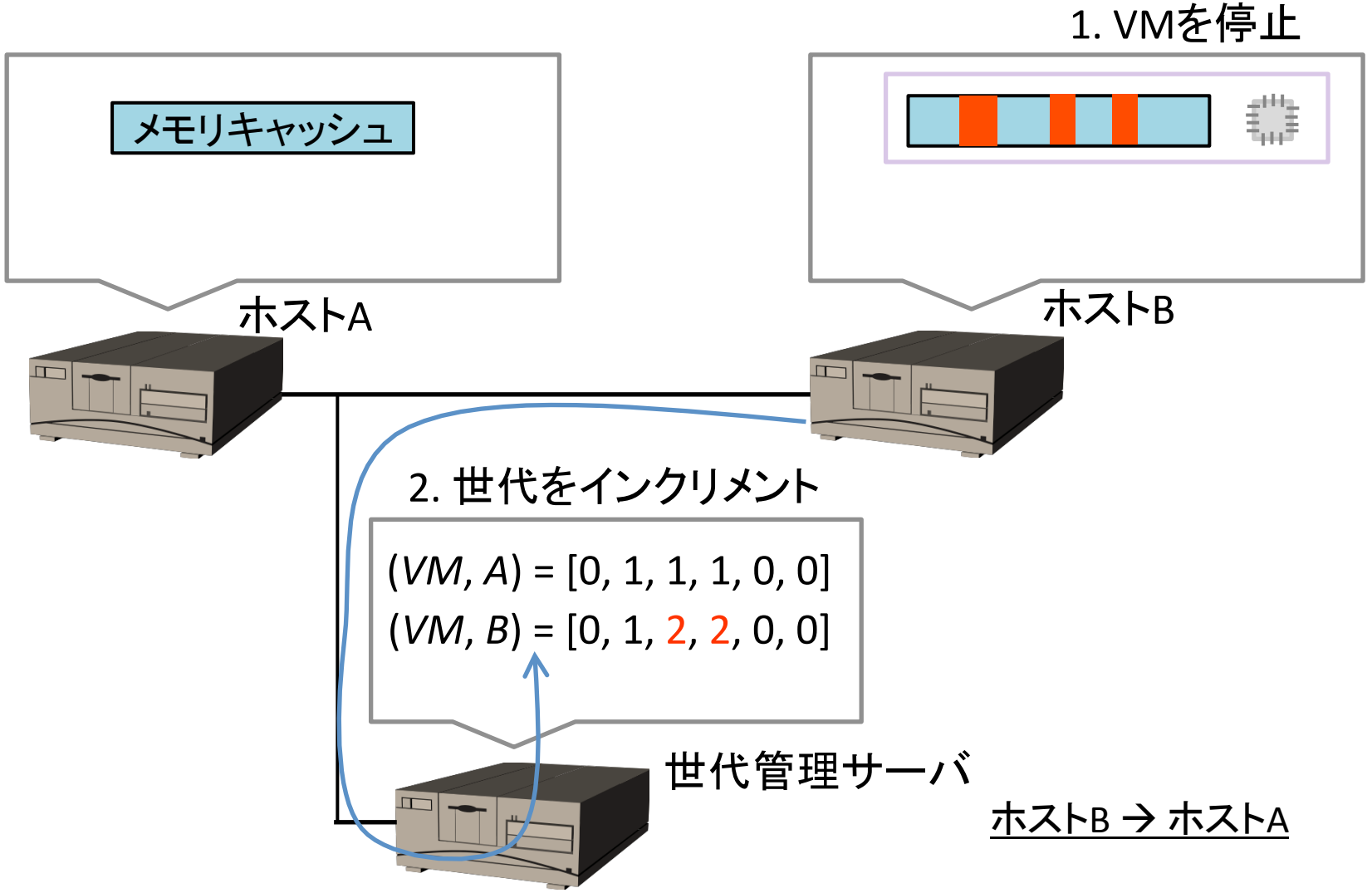
都鳥の動作: 戻るマイグレーション



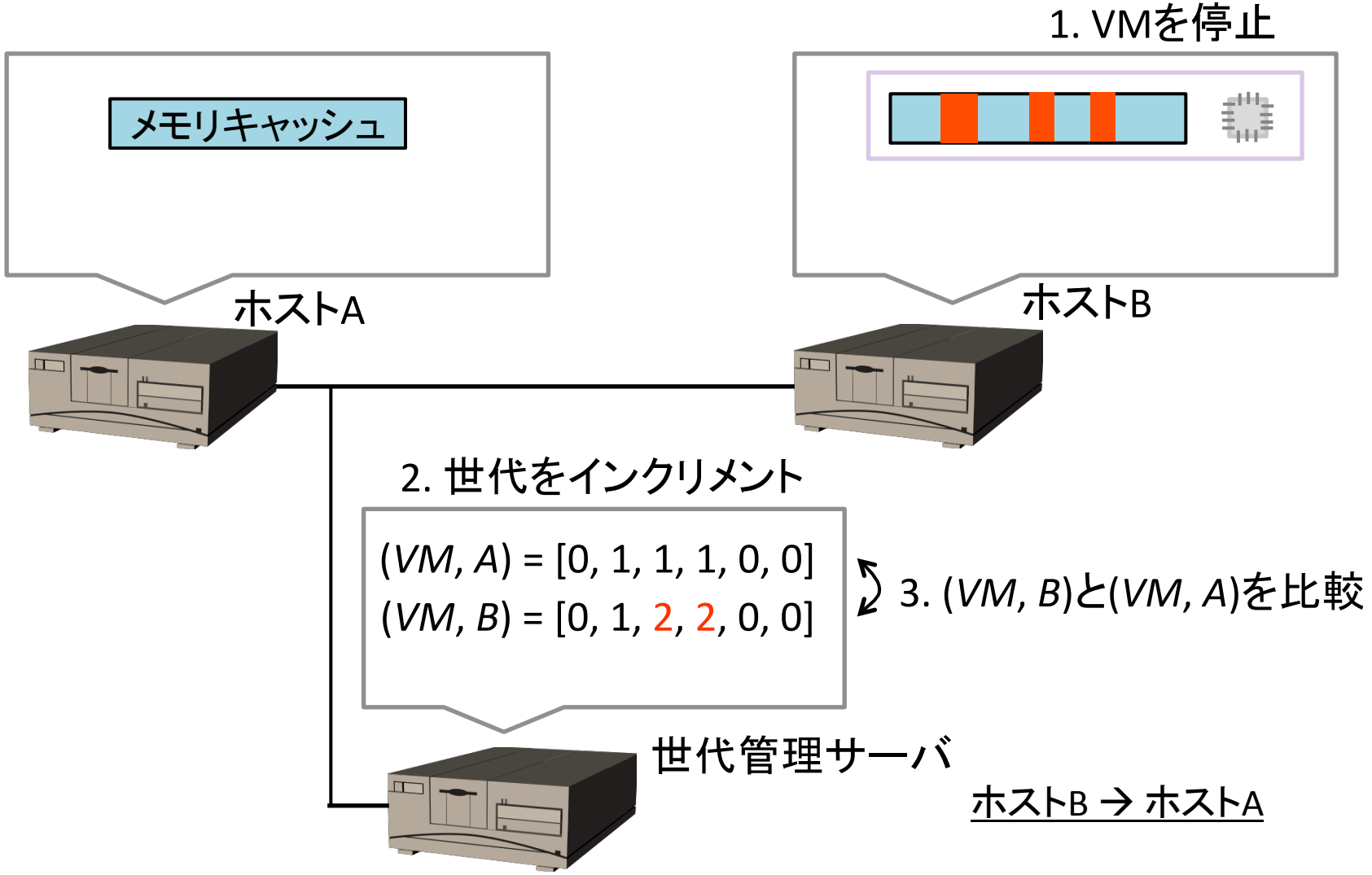
都鳥の動作: 戻るマイグレーション



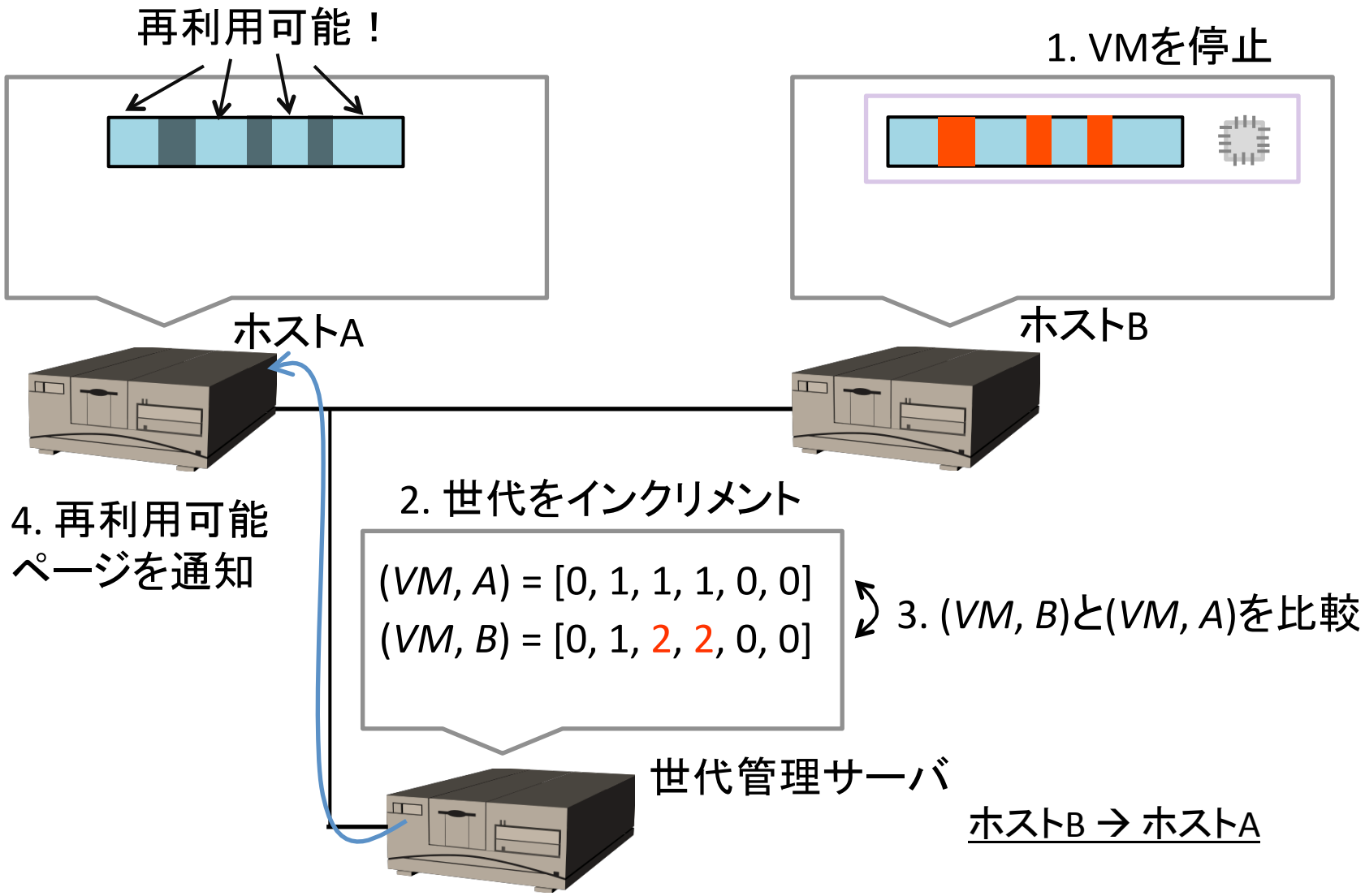
都鳥の動作: 戻るマイグレーション



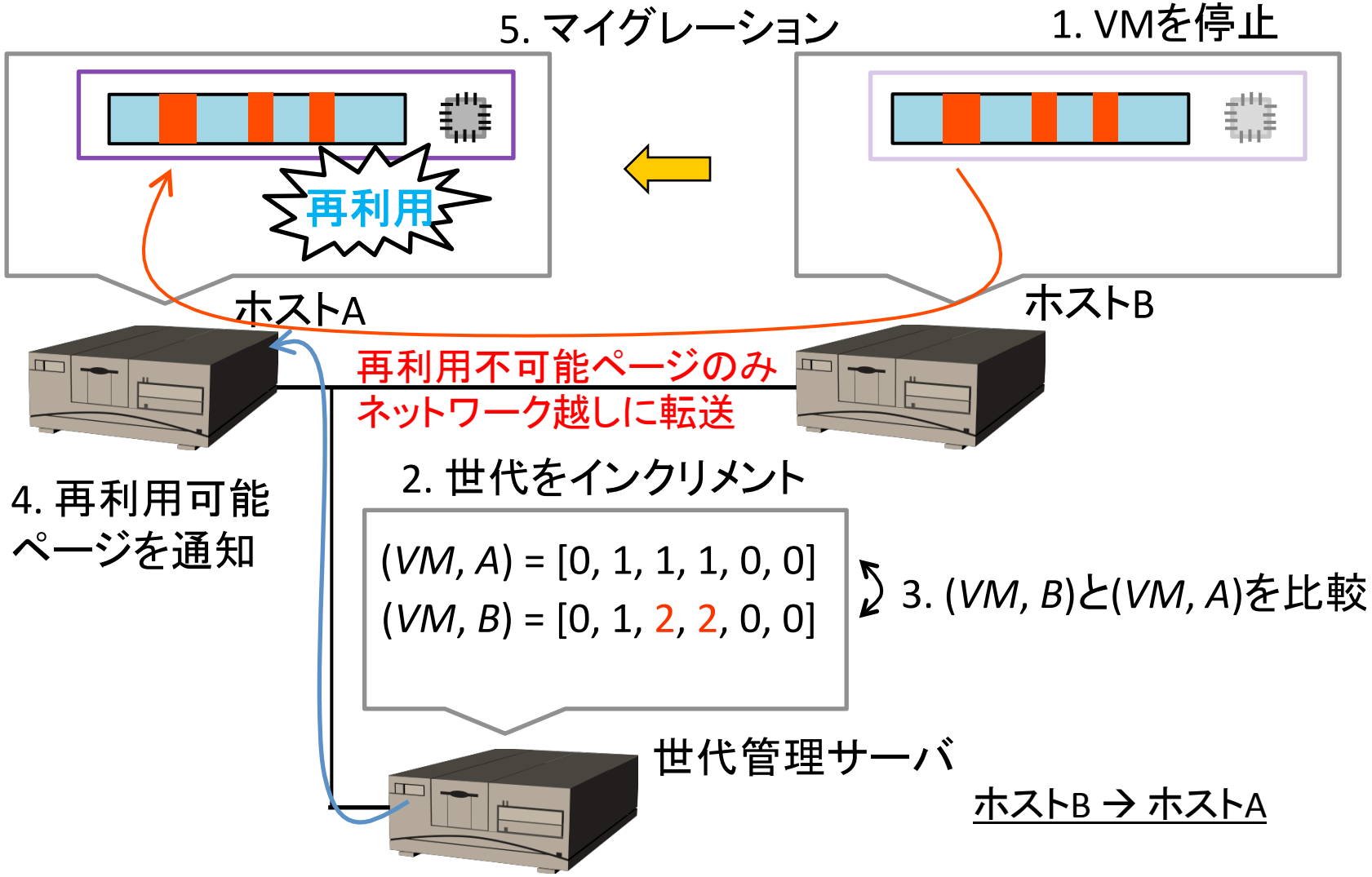
都鳥の動作: 戻るマイグレーション



都鳥の動作: 戻るマイグレーション



都鳥の動作: 戻るマイグレーション



実験: 目的と環境

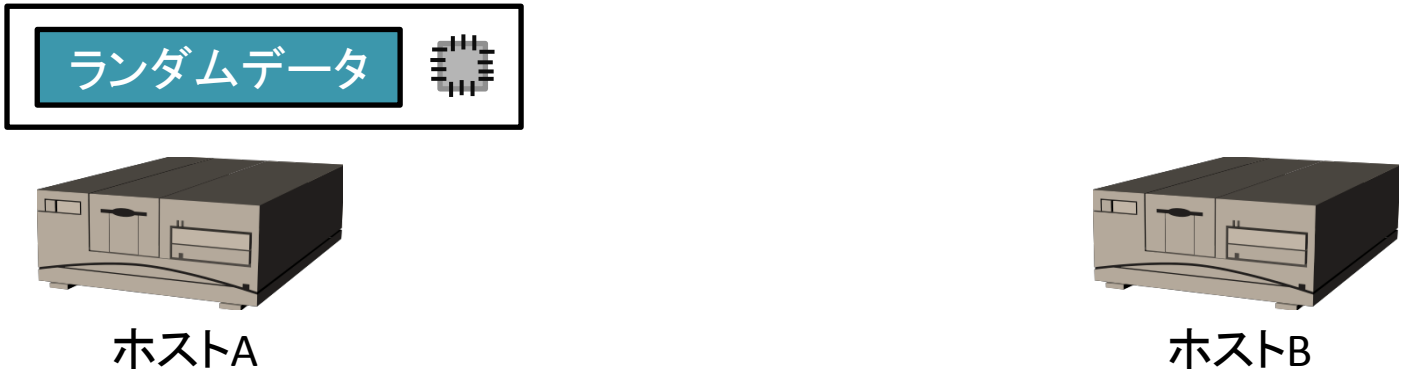
- マイクロベンチマーク (pre)
  - ページ書き込み速度と再利用率の関係
  - (Xeon X5460, 8GB memory, GibEthernet , QEMU 0.13.0, Linux 2.6.32, KVM 2.6.32)×3
- アプリケーションベンチマーク (pre/post)
  - Webサーバワークロードをシミュレート
  - (Phenom II X4 955, 4GB memory, QEMU 0.13.0, Linux 2.6.32, KVM 2.6.38)×1

KVMのバグのため

ゲストはともにUbuntu 10.10 server, 1GB memory

マイクイロベンチマーク(1/2)

1. ホストAで500MBのランダムデータを用意
2. ホストAからホストBへマイグレーション
3. データを更新（三種類の速度）
   - 200 pages/s, 1000 pages/s, 5000 pages/s
4. 30秒後にホストBからホストAに戻る

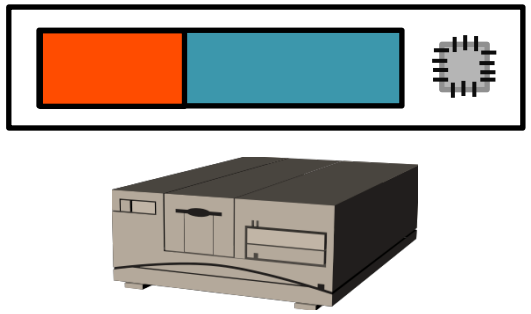


マイクイロベンチマーク(1/2)

1. ホストAで500MBのランダムデータを用意
2. ホストAからホストBへマイグレーション
3. データを更新（三種類の速度）
   - 200 pages/s, 1000 pages/s, 5000 pages/s
4. 30秒後にホストBからホストAに戻る

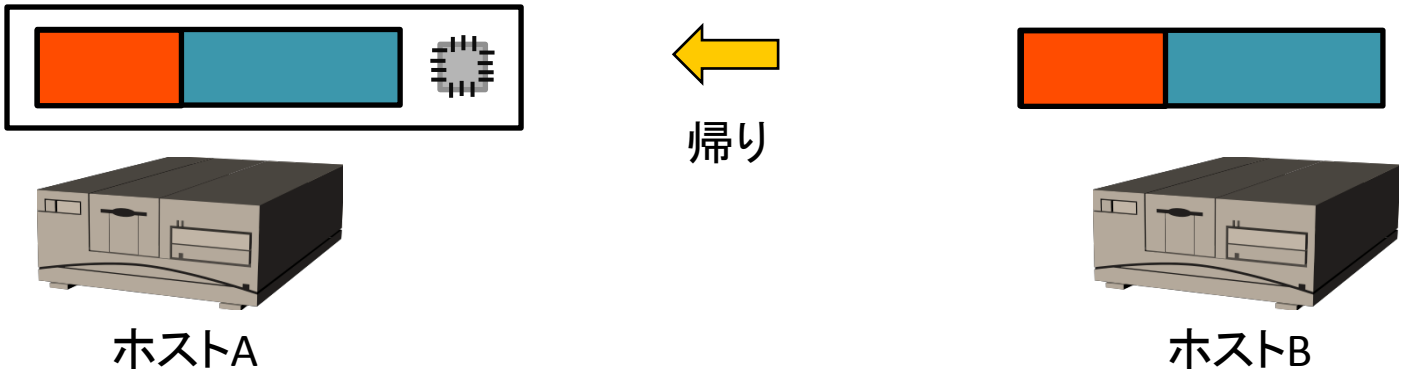
マイクイロベンチマーク(1/2)

1. ホストAで500MBのランダムデータを用意
2. ホストAからホストBへマイグレーション
3. データを更新（三種類の速度）
   - 200 pages/s, 1000 pages/s, 5000 pages/s
4. 30秒後にホストBからホストAに戻る

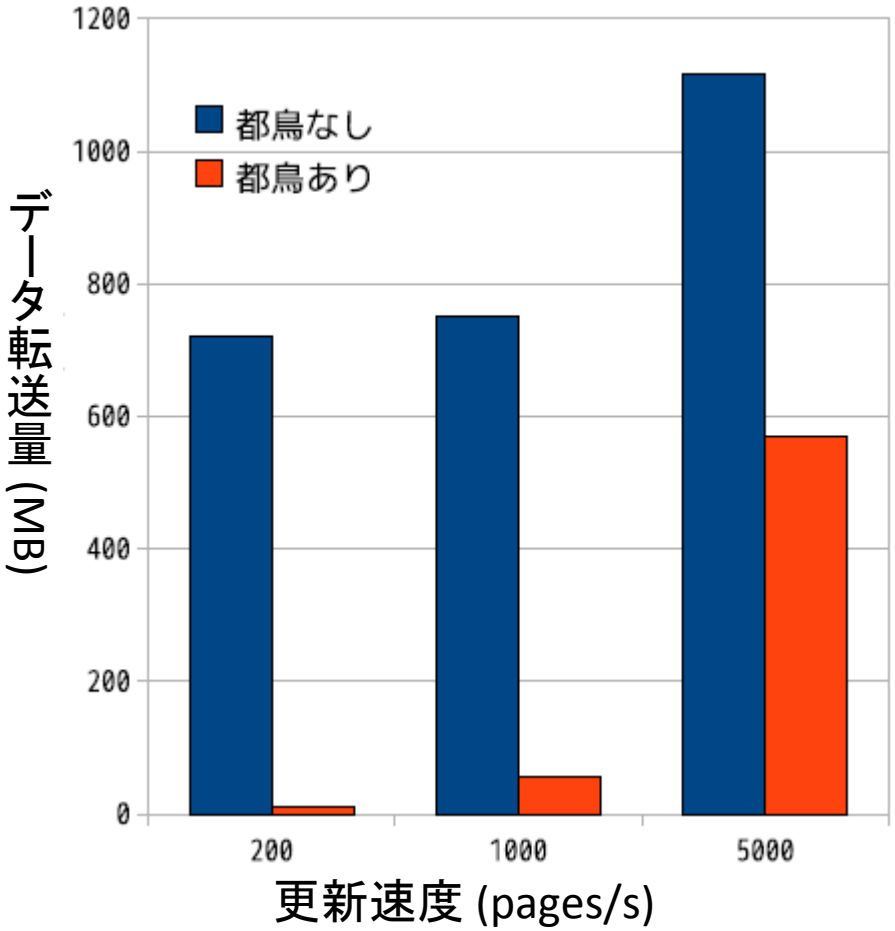


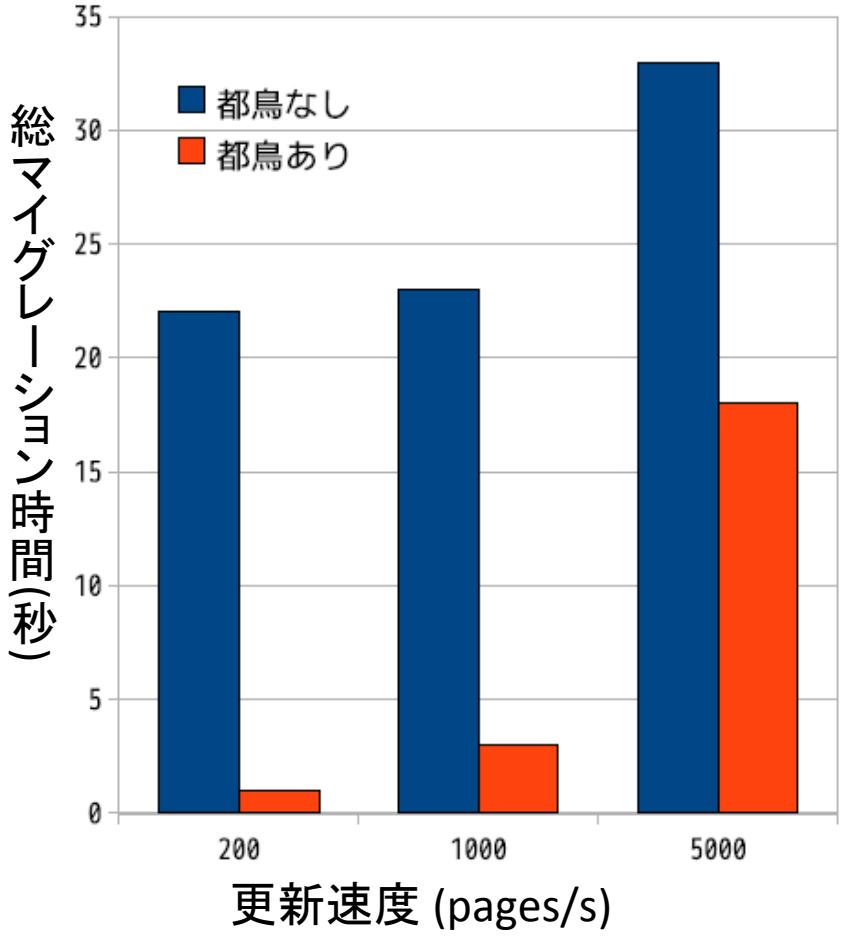
マイクイロベンチマーク(1/2)

1. ホストAで500MBのランダムデータを用意
2. ホストAからホストBへマイグレーション
3. データを更新（三種類の速度）
   - 200 pages/s, 1000 pages/s, 5000 pages/s
4. 30秒後にホストBからホストAに戻る



マイクロベンチマーク(2/2)

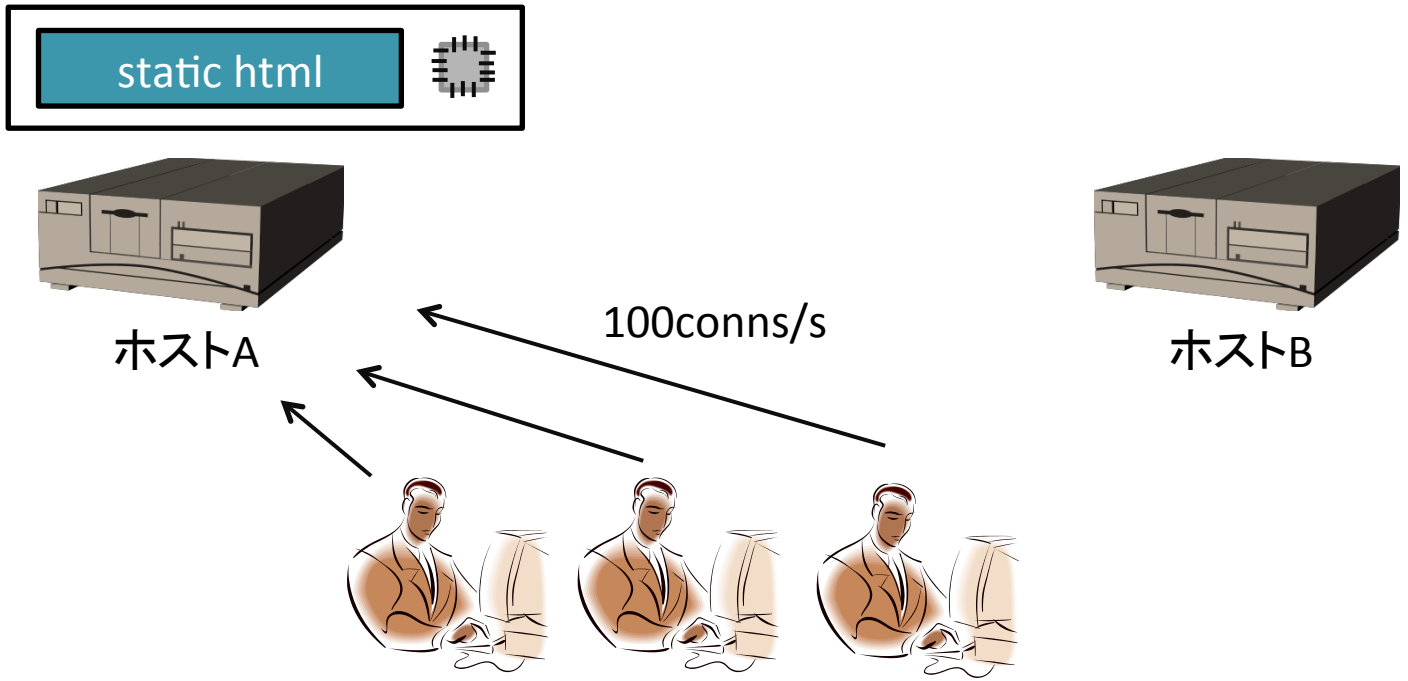




- 低速更新時: 転送量、移動時間共に9割以上の削減
- 高速更新時: 削減率が下がるものの5割ほど削減

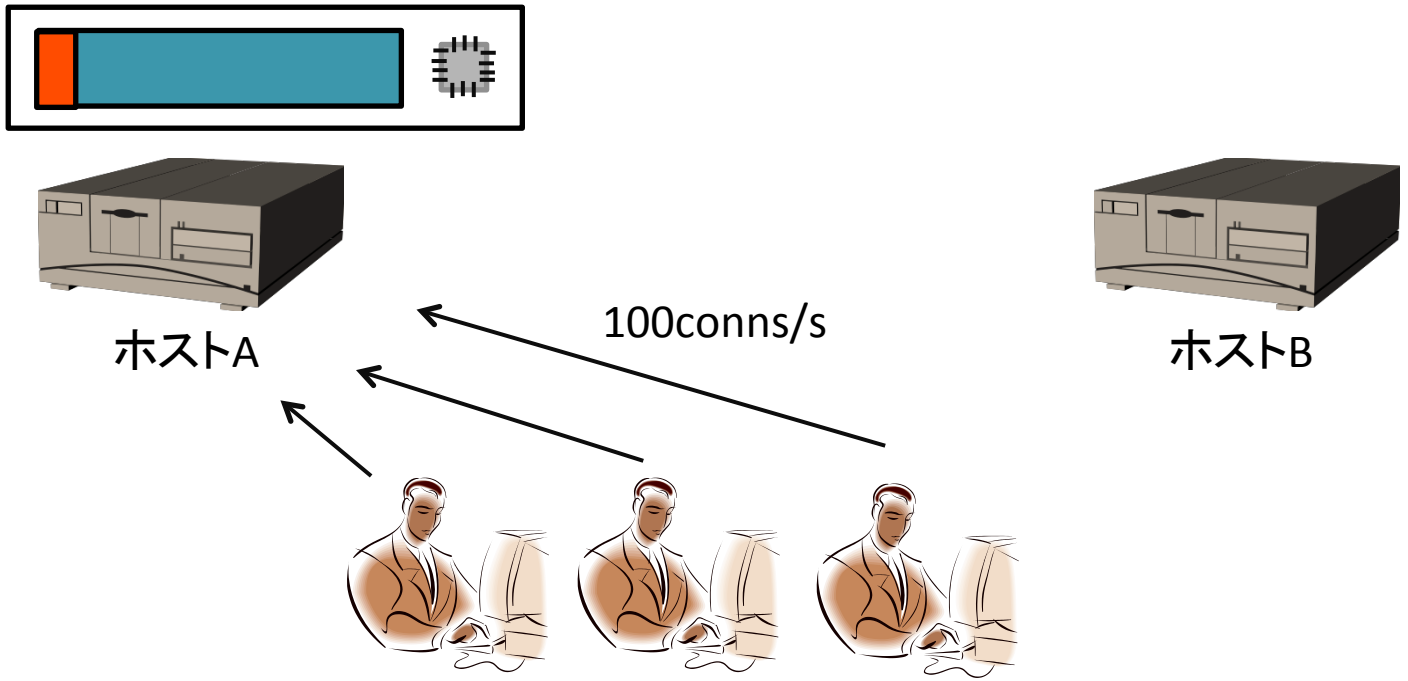
アプリケーションベンチマーク(1/2)

1. 256KiB ×1024個の静的htmlをapacheで公開
2. 100接続/sで10周読み込み（約100秒）
3. 30秒後にA→B、さらに30秒後B→A



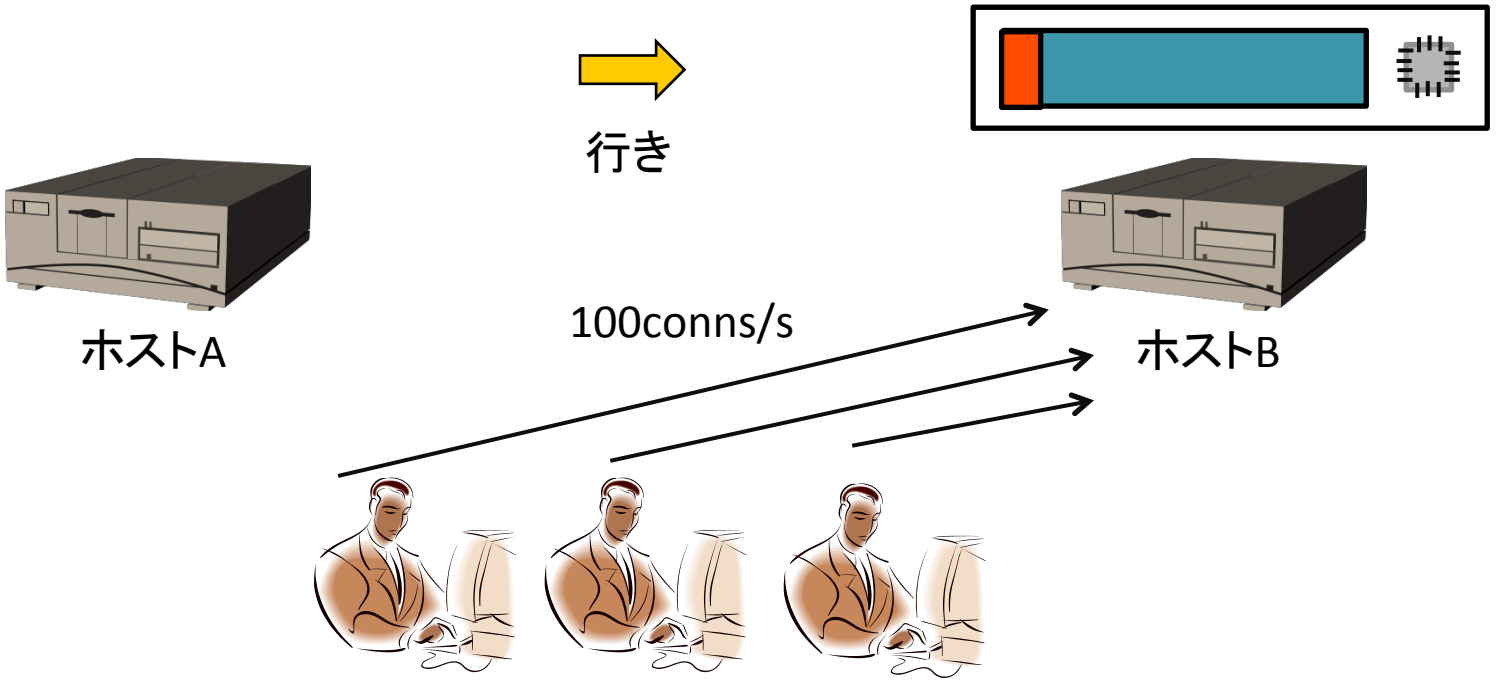
アプリケーションベンチマーク(1/2)

1. 256KiB ×1024個の静的htmlをapacheで公開
2. 100接続/sで10周読み込み（約100秒）
3. 30秒後にA→B、さらに30秒後B→A



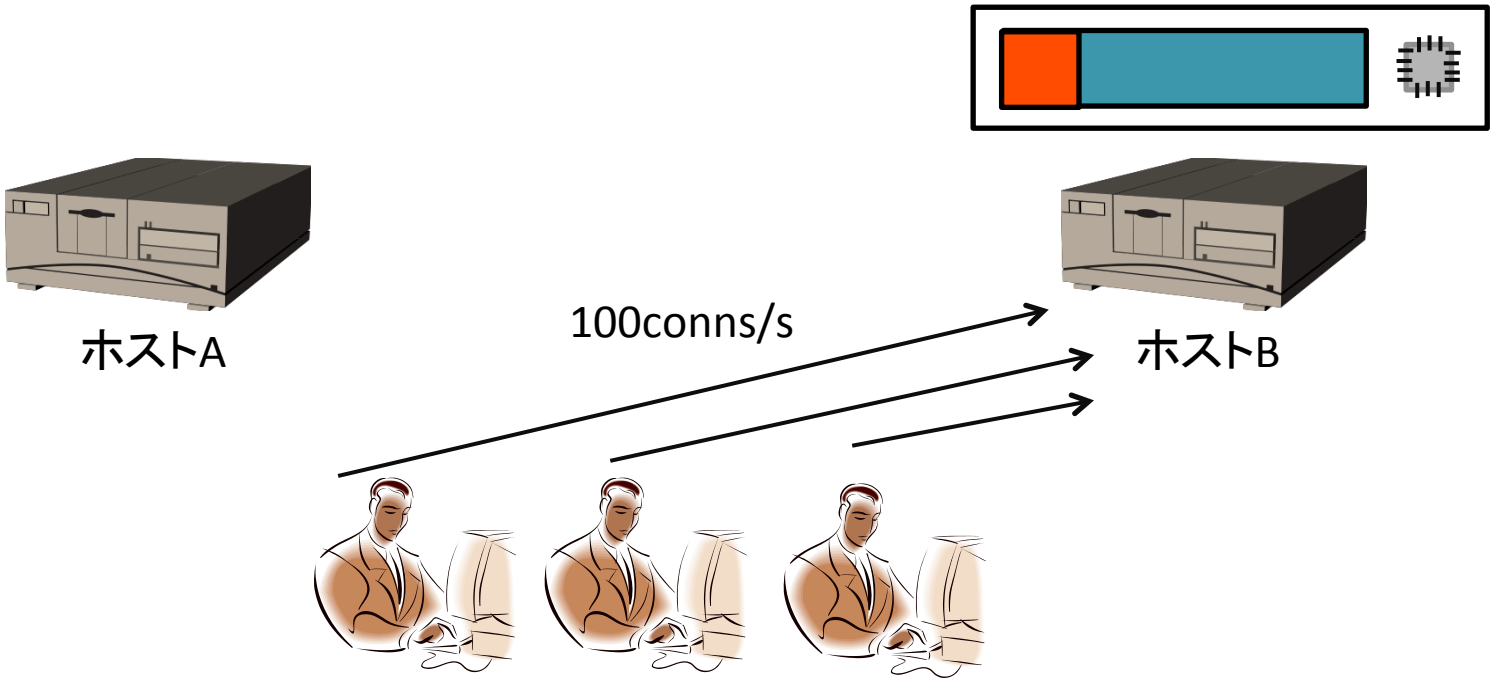
アプリケーションベンチマーク(1/2)

1. 256KiB ×1024個の静的htmlをapacheで公開
2. 100接続/sで10周読み込み（約100秒）
3. 30秒後にA→B、さらに30秒後B→A



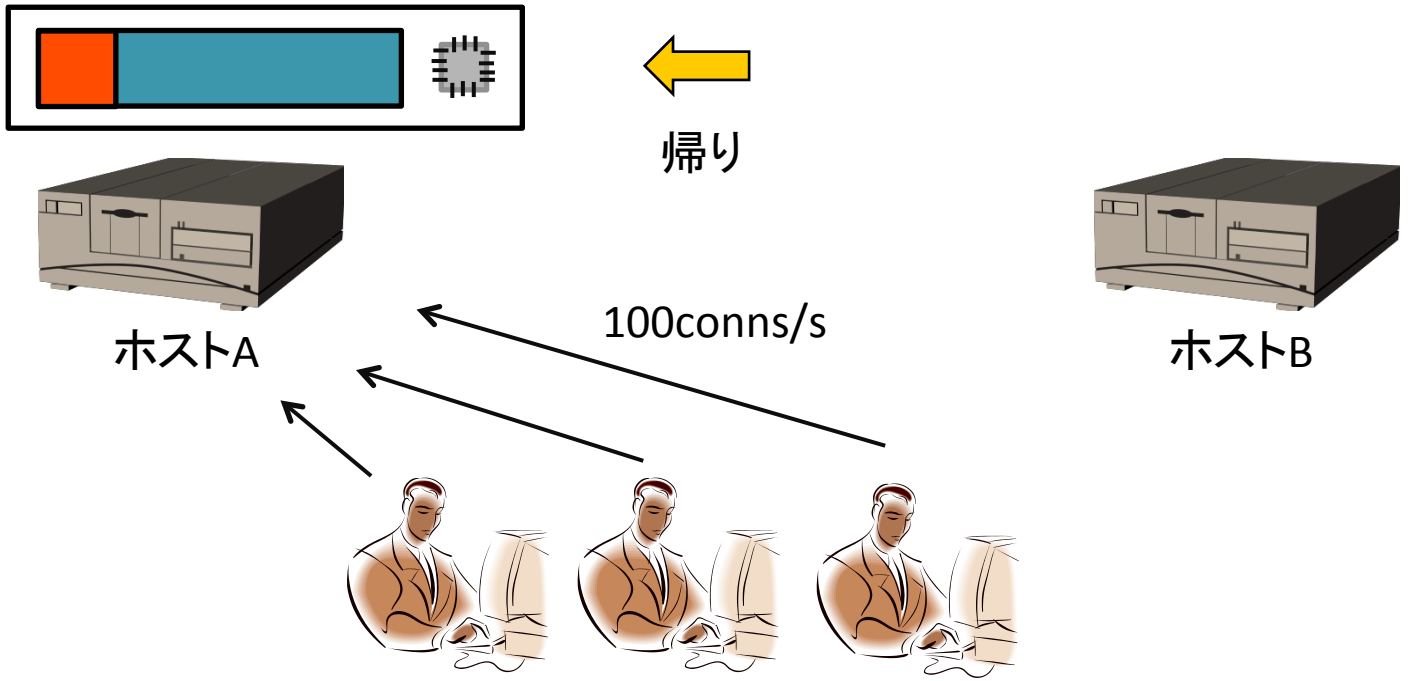
アプリケーションベンチマーク(1/2)

1. 256KiB ×1024個の静的htmlをapacheで公開
2. 100接続/sで10周読み込み（約100秒）
3. 30秒後にA→B、さらに30秒後B→A

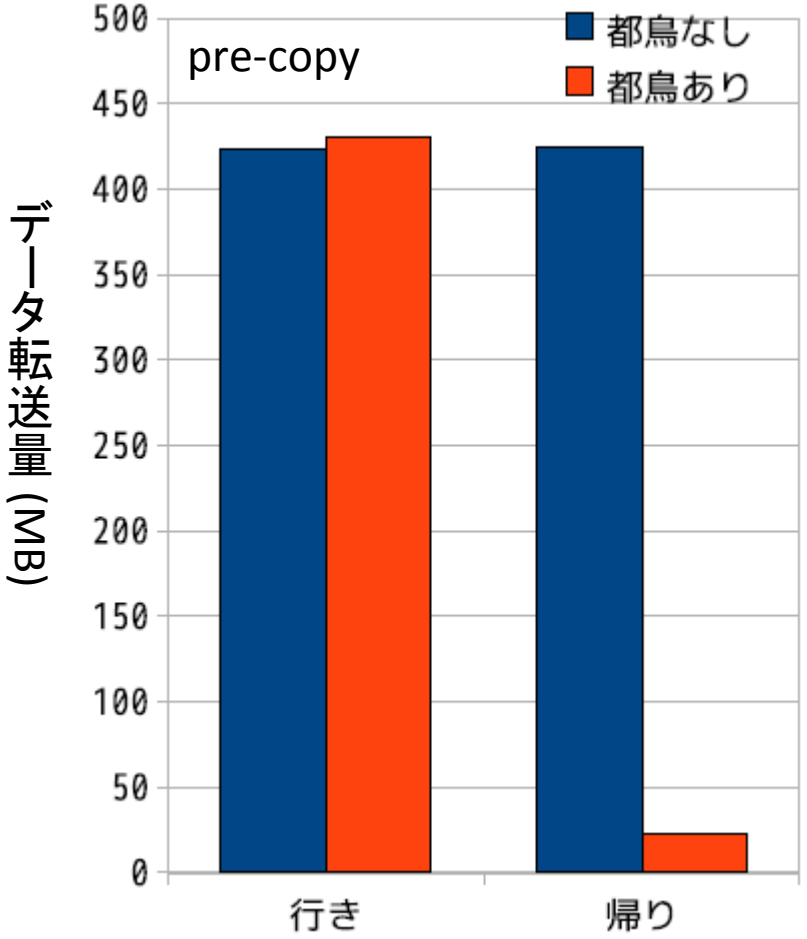


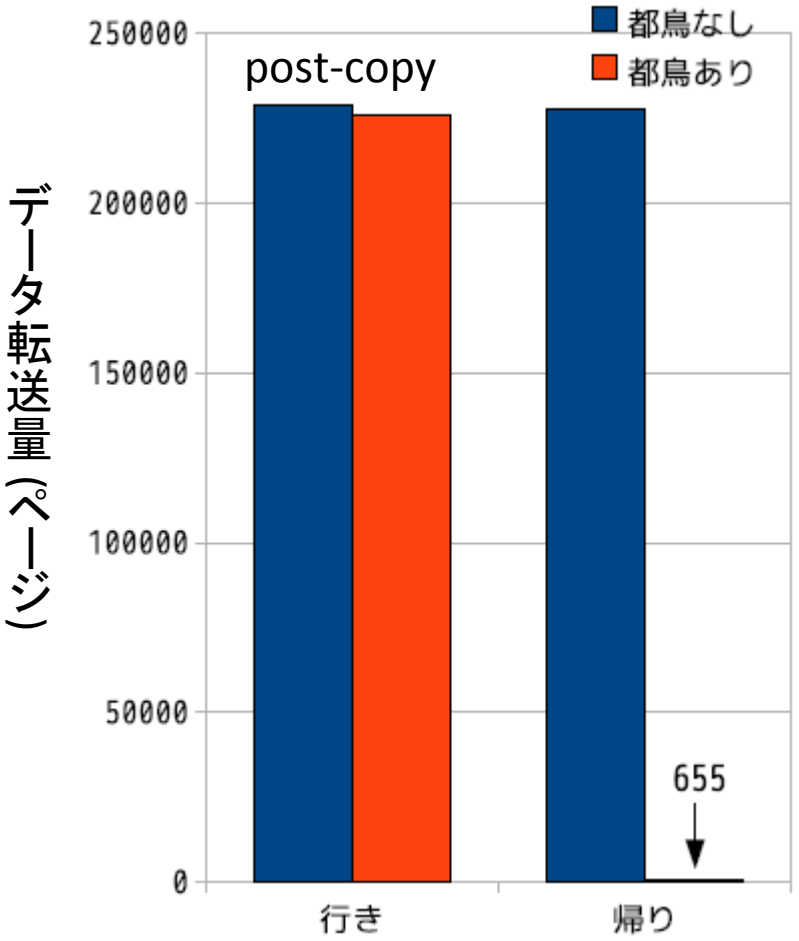
アプリケーションベンチマーク(1/2)

1. 256KiB ×1024個の静的htmlをapacheで公開
2. 100接続/sで10周読み込み（約100秒）
3. 30秒後にA→B、さらに30秒後B→A



アプリケーションベンチマーク(2/2)





- 9割以上の削減
- ファイル読込はメモリを更新しない（手法の適用可能領域）

関連研究

- 単発のライブマイグレーションに関する研究
  - pre-copy型の繰り返しコピーで差分のみをrun-lengthで圧縮して転送 [9]
  - メモリではなく実行ログを転送→転送量削減 [10]
  - メモリページを特性によってクラスタリングしpost-copyにおけるプリフェッチ精度を向上 [11]
- 本研究は繰り返し発生するライブマイグレーションに着目した点が異なる

[9] Evaluation of Delta Compression Techniques for Efficient Live Migration of Large Virtual Machines, *VEE2011*, Svärd *et al*.

[10] Live Migration of Virtual Machine Based on Full System Trace and Replay, *HPDC2009,* Liu *et al.*

[11] Kaleidoscope: Cloud Micro-Elasticity via VM State Coloring, *EuroSys*2011, Bryant *et al*.

今後の課題

1. メモリキャッシュの効率的な管理
   - 1GBのVMを実行したホスト → 当該VMのメモリキャッシュに1GBのRAMを消費
   - 単純にはあるホストで更新されたページはその他のホストのキャッシュから解放可能(∵再利用できない)
2. 実環境での評価
   - 負荷に応じたサーバ集約との統合評価
   - ワークロード？(検討中)

結論

- 従来のライブマイグレーションでは全メモリを転送 → 多くの転送量, 長い転送時間
- VMの積極的な再配置による最適化では問題
- VMのメモリを再利用し上記を削減
  - 実装(都鳥)により有用性が示された
- 今後の課題
  - 効率的なキャッシュ管理
  - 実環境での評価

ComSys2011でのQ&A

- 提案手法のオーバーヘッドは？
  - メモリ書き込み監視によるオーバーヘッドは実用上十分小さい（NPBで確認）
  - 移動元・移動先・世代管理サーバ間で通信するが、(メモリサイズ÷4K)ビットと小さい
- キャッシュを先に転送しておくことは可能か？
  - 可能。負荷予測が必要なため現状out of scope
- どのようなワークロードに適用？
  - ファイルキャッシュが大きいとよく効く。具体的には現在検討中。