準仮想化ページフォルトを用いたポストコピー型ライブマイグレーションの性能向上手法

広渕崇宏*　山幡為佐久**　伊藤智*

* 産業技術総合研究所

** VA Linux Systems Japan

コンピュータシステムシンポジウム2012
2012年12月6日

本スライドは2012年12月当時のものです。
その後、実装を改良し包括的な性能評価を行いました。下記の論文をご参照ください。
Qemu 2.5 + Linux Kernel 4.3以降においては、
ポストコピー型ライブマイグレーションの機能が標準的に組み込まれています。

Postcopy Live Migration with Guest-cooperative Page Faults,
Takahiro Hirofuchi, Isaku Yamahata, Satoshi Itoh,
IEICE Transactions on Information and Systems, pp.2159-2167, Vol.E98-D, No.12, IEICE, Dec 2015
DOI: 10.1587/transinf.2015PAP0011
PDF (Full Text)

背景

- プレコピー型ライブマイグレーション
  - 今日一般的に使用されている方式
  - メモリを宛先に転送してから実行ホストを切り替え
- ポストコピー型ライブマイグレーション
  - 実行ホストを切り替えてからメモリを転送
  - 短時間かつ必ず一定時間で移動が完了する
    - 積極的なサーバコンソリデーションで省エネ
    - ハードウェアメンテナンスを容易化
  - 「Yabusame」をQemu/KVM向けに開発中
  - でも、性能低下してしまう場合がある
    - VMが未転送のメモリページにアクセスすると、ゲストOSを一時的に止めないといけない

本研究の貢献

- ポストコピー型ライブマイグレーションの性能低下を緩和する手法を提案
  - VMが未転送のメモリページにアクセスしたら、ゲストOS全体を止めるのではなく、そのページにアクセスしたゲストOS上のプロセスのみ止め、他のプロセスの実行は継続する仕組み
- 評価の結果、性能低下の緩和効果が確認できた

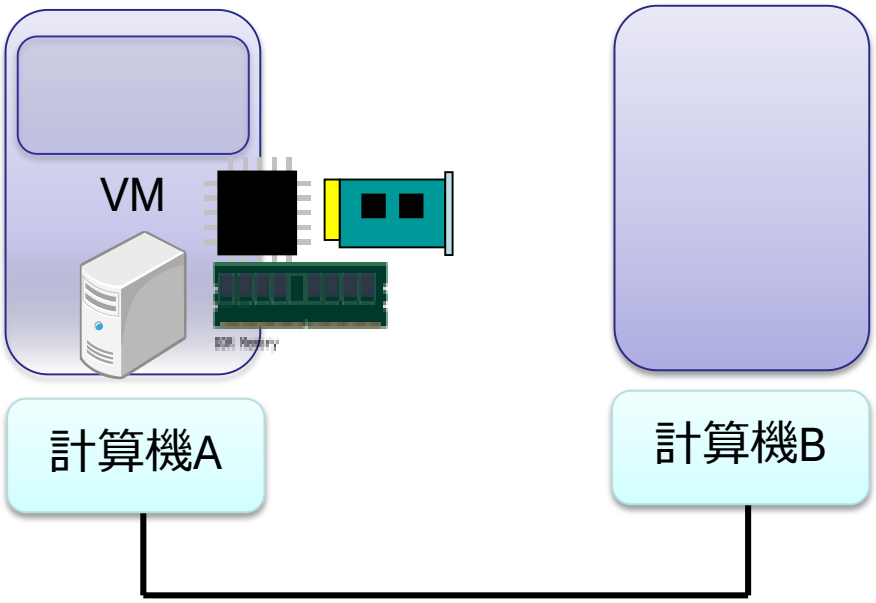
発表のながれ

- ポストコピー型ライブマイグレーション
  - 動作原理
  - 本研究で取り組む課題
- 提案手法
  - 設計
  - 動作概要
  - 実装
- 評価
  - 評価実験
- 関連研究
- 開発状況
- まとめ

ポストコピー型の動作概要（１）

実行ホストを切り替えた後にメモリーをコピー

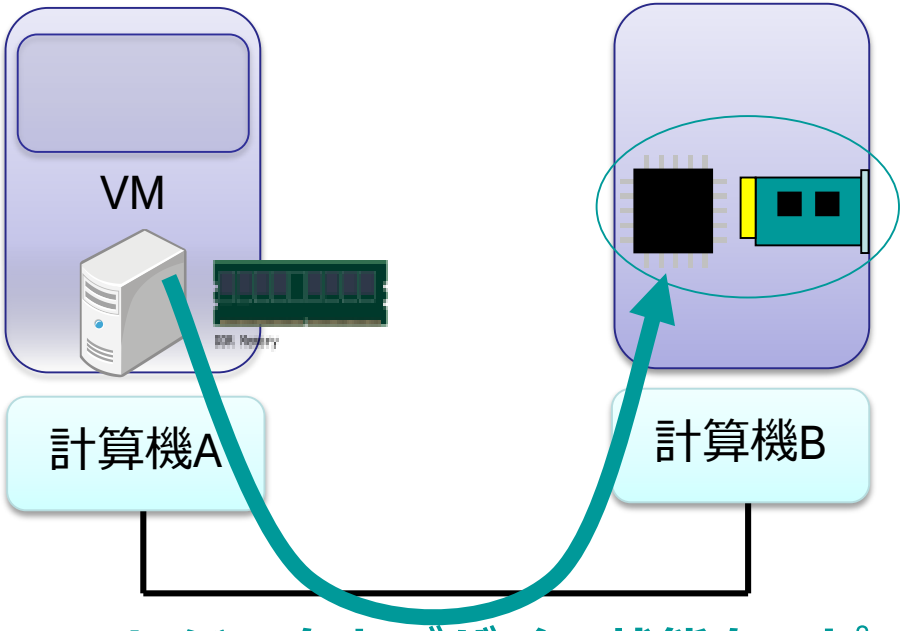
停止



1. 移動元でVMを停止
2. レジスタとデバイス状態を宛先にコピー
3. 宛先でVM実行を再開
4. 必要なメモリページをオンデマンドに取得

ポストコピー型の動作概要（２）

実行ホストを切り替えた後にメモリーをコピー

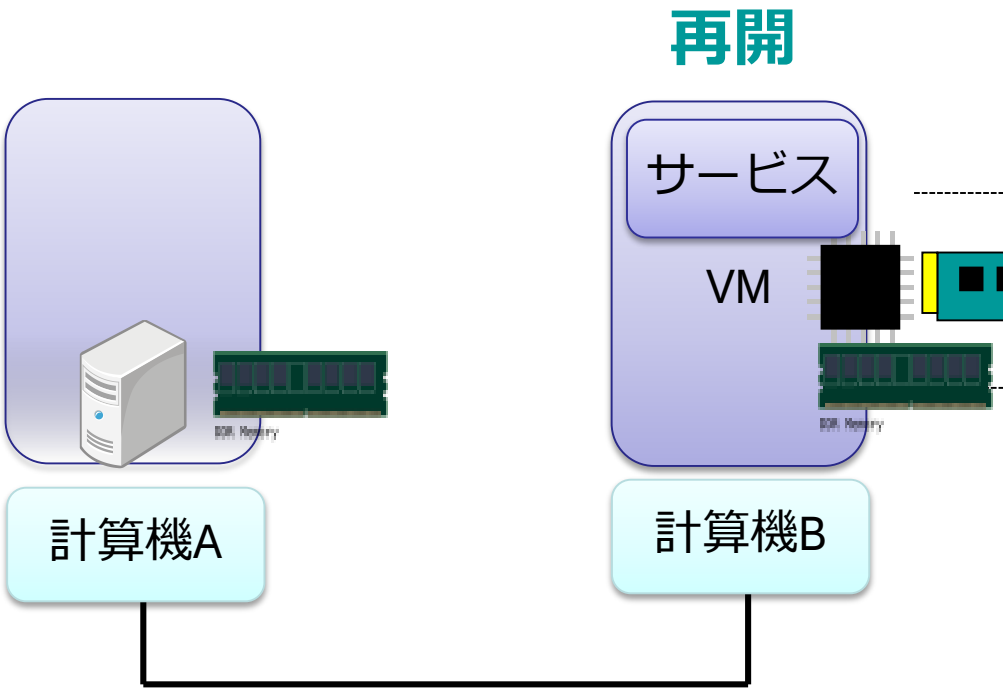


1. 移動元でVMを停止
2. CPUレジスタとデバイス状態を宛先にコピー
3. 宛先でVM実行を再開
4. 必要なメモリページをオンデマンドに取得

CPUレジスタとデバイス状態をコピー
VGAデバイス無しならたった256KB
=> ほぼ瞬間的な実行ホスト切り替えが可能

ポストコピー型の動作概要（３）

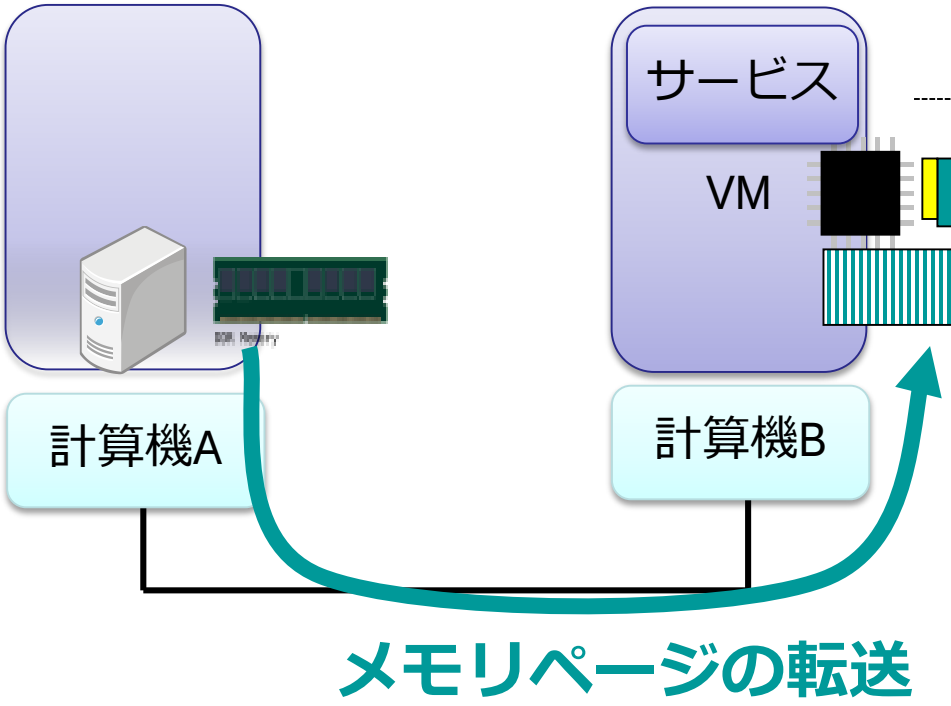
実行ホストを切り替えた後にメモリーをコピー



1. 移動元でVMを停止
2. CPUレジスタとデバイス状態を宛先にコピー
3. 宛先でVM実行を再開
4. 必要なメモリページをオンデマンドに取得

ポストコピー型の動作概要（４）

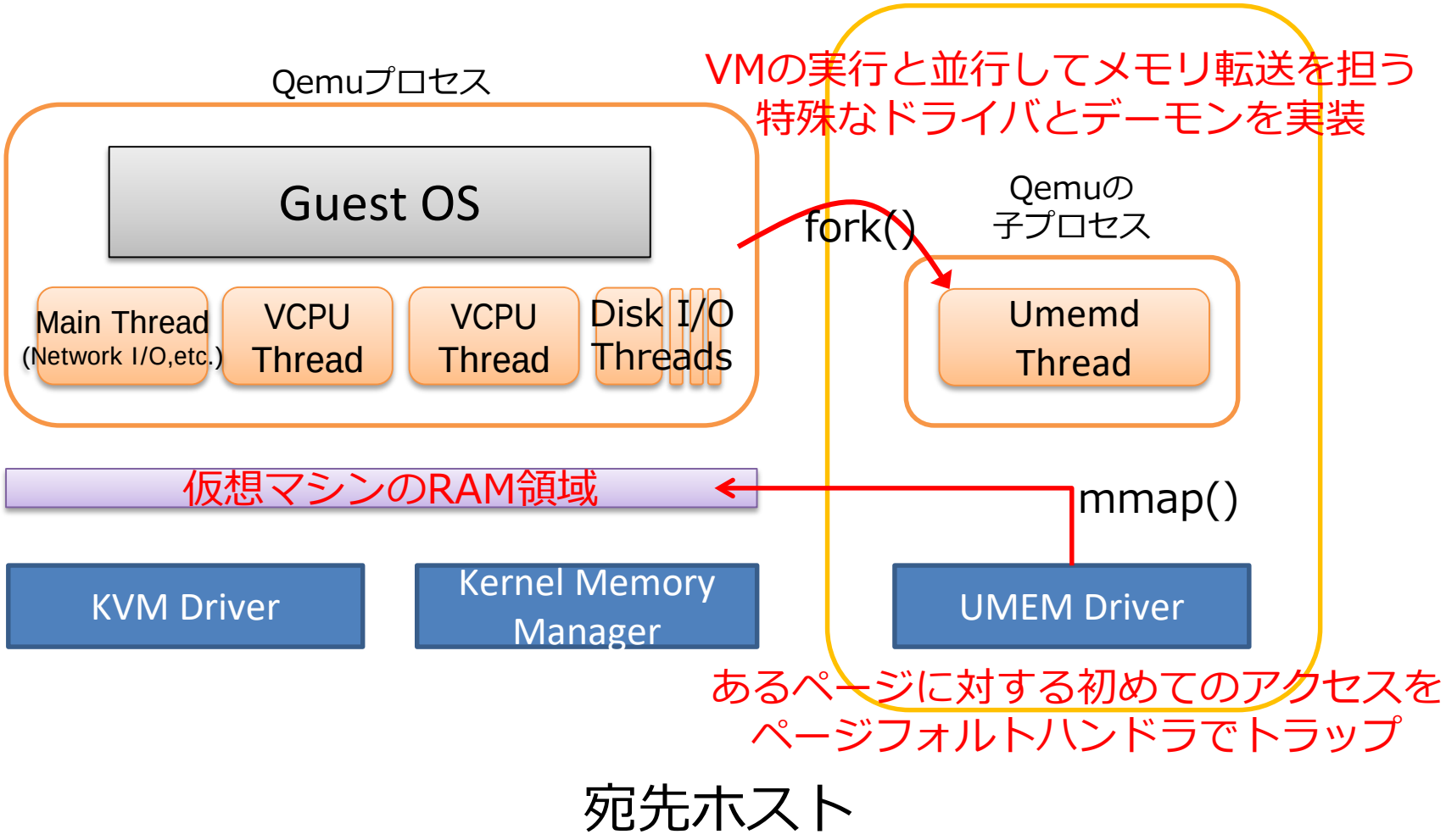
実行ホストを切り替えた後にメモリーをコピー



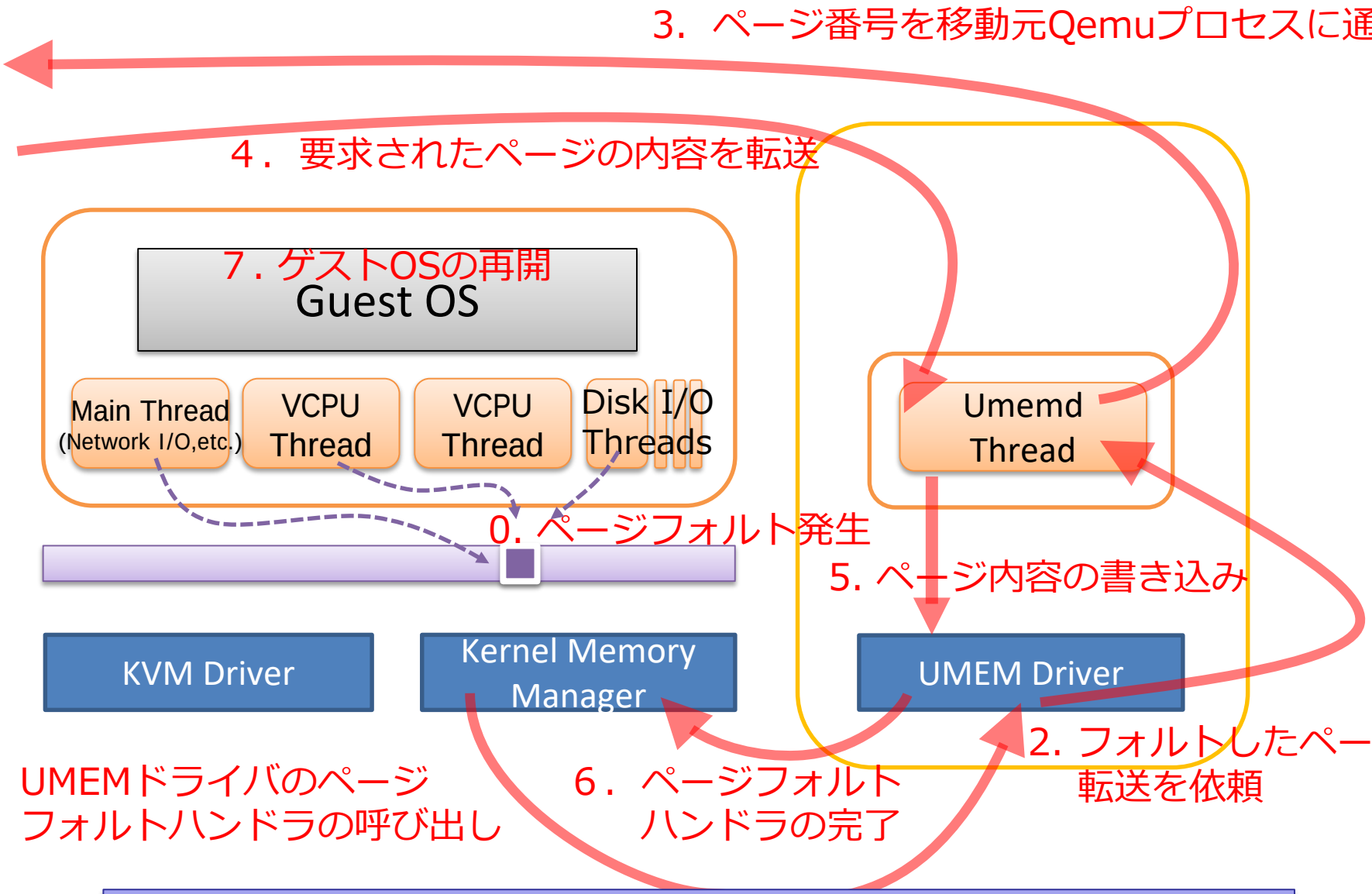
1. 移動元でVMを停止
2. CPUレジスタとデバイス状態を宛先にコピー
3. 宛先でVM実行を再開
4. 必要なメモリページをオンデマンドに取得

並行して残りも転送

Qemu/KVMにおけるポストコピー型ライブマイグレーションの実装（Yabusame）



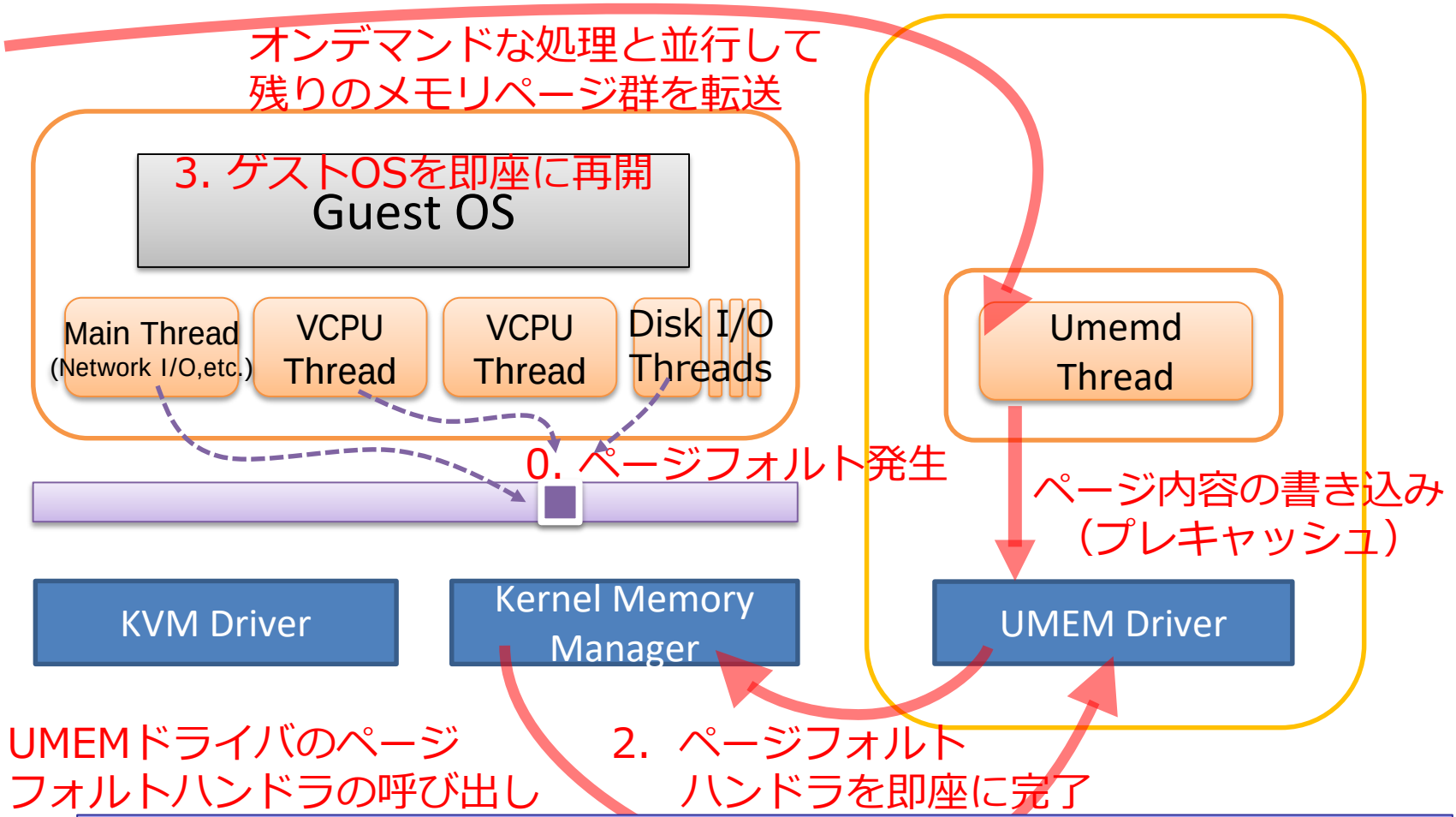
オンデマンドなページ転送



オンデマンドなページ転送の処理中はゲストOSが停止してしまう。

バックグラウンドでのページ転送

既存の性能低下緩和手法



バックグラウンドでのページ転送は、アクセス頻度が高いメモリ領域を先に転送して、オンデマンドな転送の発生頻度を抑制する。

本研究で取り組む問題

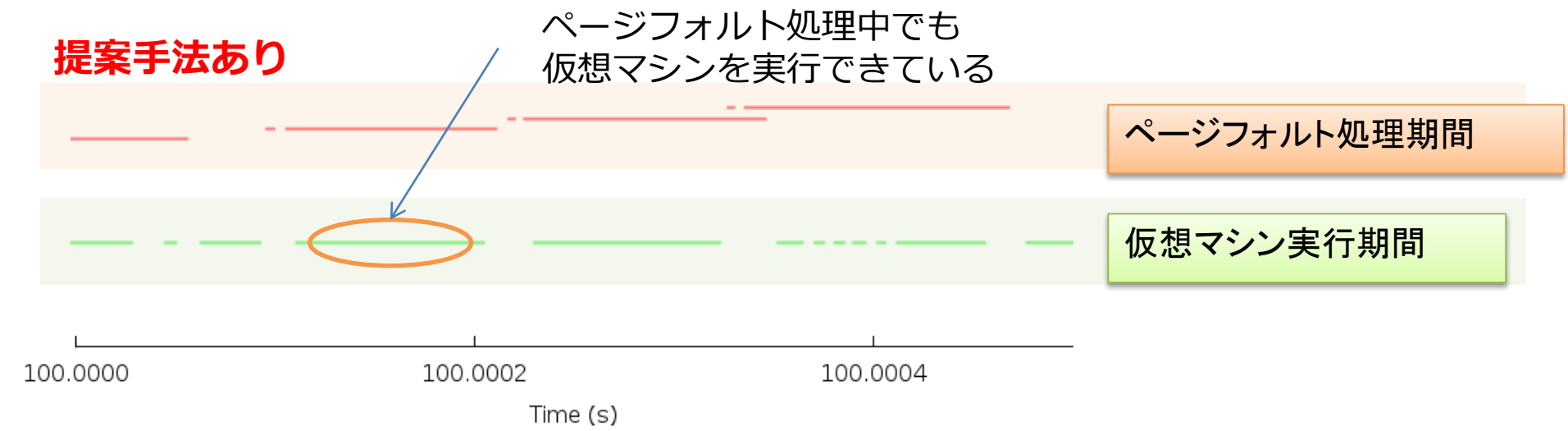
- バックグラウンドでの転送（プレキャッシュ）は一定の効果があるものの、依然としてキャッシュミスによる性能低下の可能性が残る
  - メジャーフォルトの一回あたりの停止時間はごくわずか（GbEで数ms）だが、頻出すると大きく性能が低下する
  - キャッシュミスがある程度起きるのはしょうがないので、起きてしまったときに緩和できる方法を考える

提案手法

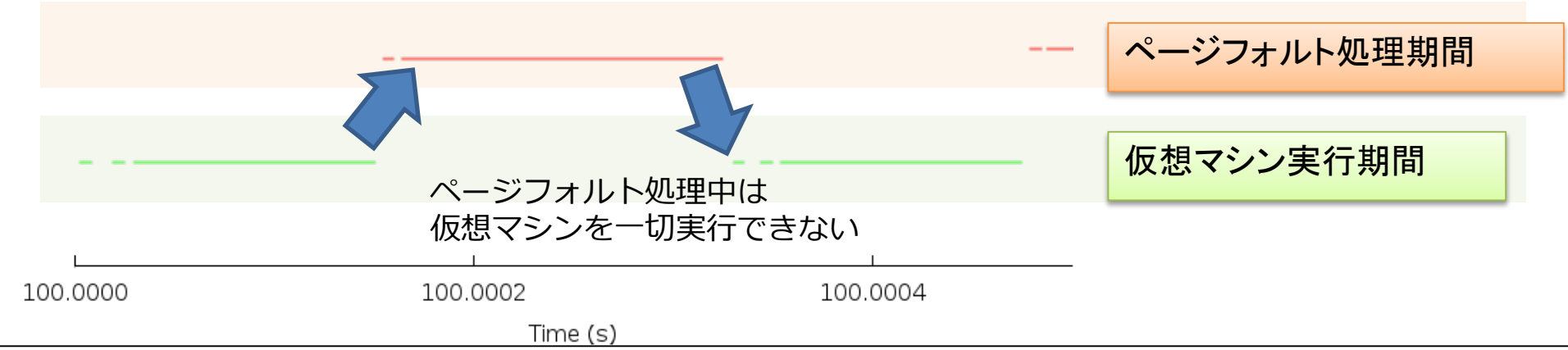
- オンデマンドなページ転送を、VMの実行に対して並行動作可能にすることで性能低下を抑制する
  - VMが未転送のメモリページにアクセスした場合には、ゲストOSに特殊な割り込みを投入し、ゲストOSを即座に再開する
  - ゲストOSのプロセススケジューラはカレントプロセスの実行のみを停止し、他のプロセスの実行は継続する
    - 転送が完了するまで当該プロセスの実行のみを遅延
- 新たに追加する割り込み
  - Page Not Present
    - KVM: そのページは今から転送するからちょっと待って！
    - ゲストOSカーネル: それまで他のプロセス実行しよう。
  - Page Present
    - KVM: そのページの転送が完了したよ！
    - ゲストOSカーネル：そのページ待ちだったプロセス再開しよう。

実行例
（SystemTapによる分析）

提案手法あり

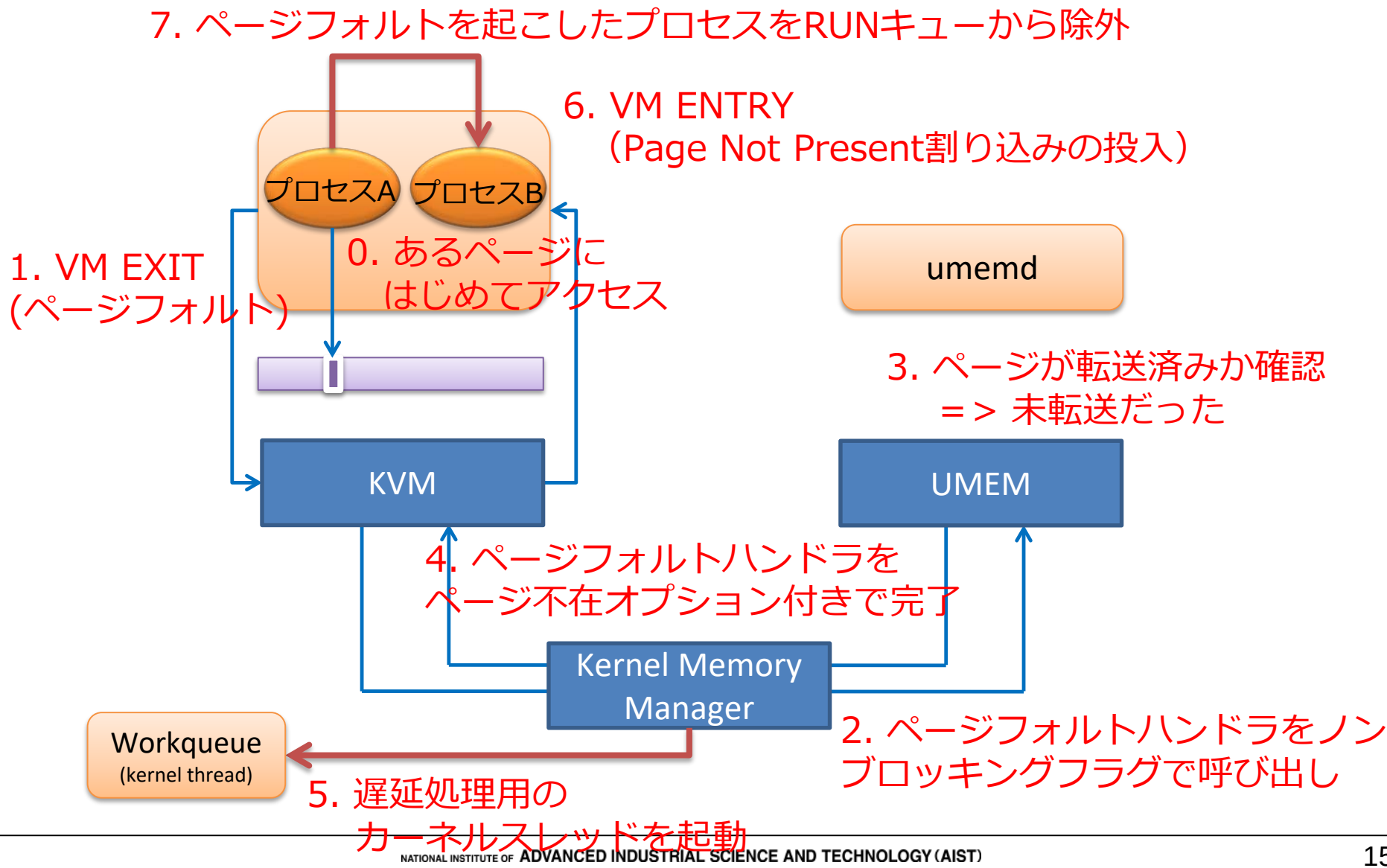


提案手法なし

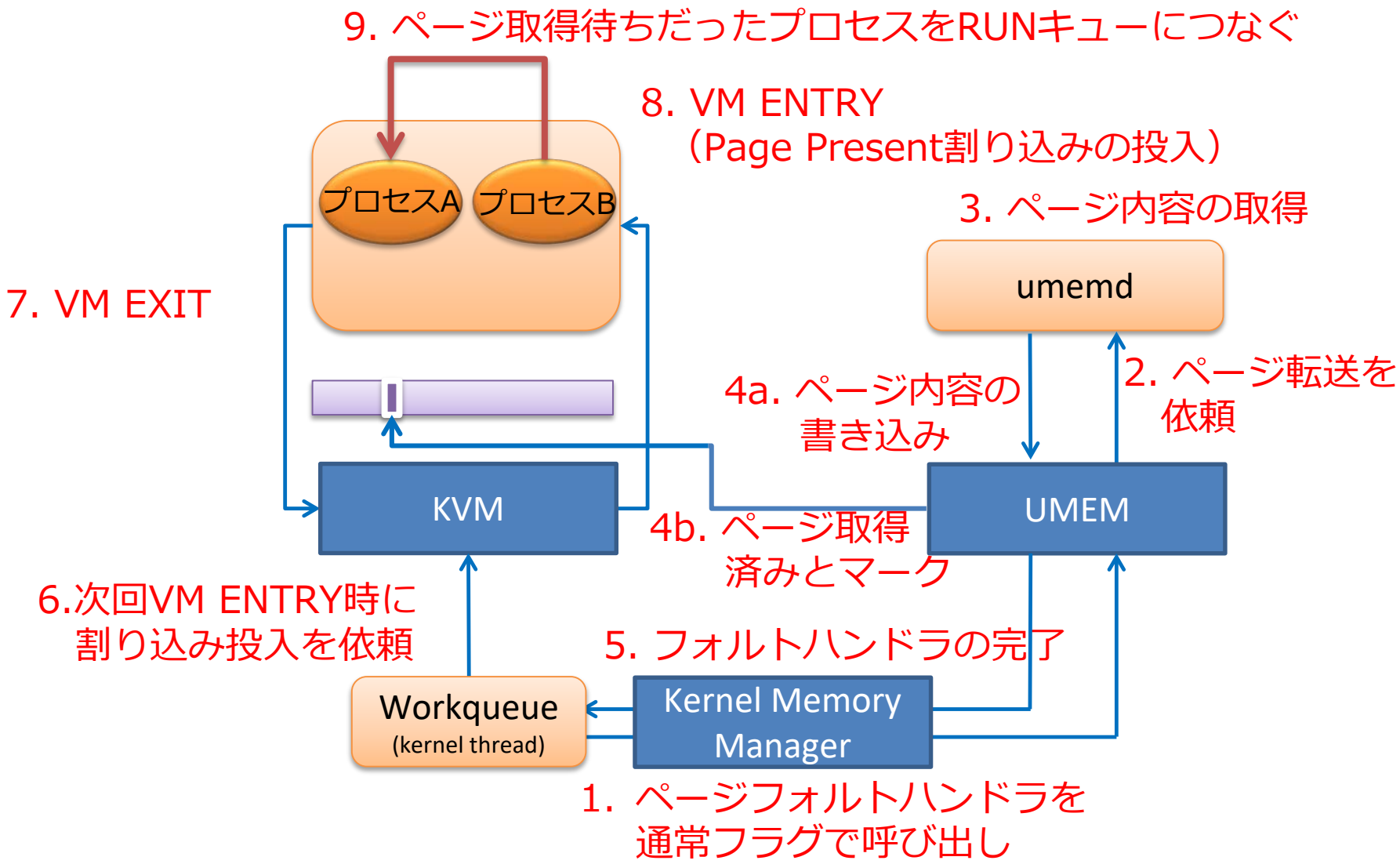


提案機構の動作概要

（Page Not Present割り込みの投入）



提案機構の動作概要
（Page Present割り込みの投入）



実装

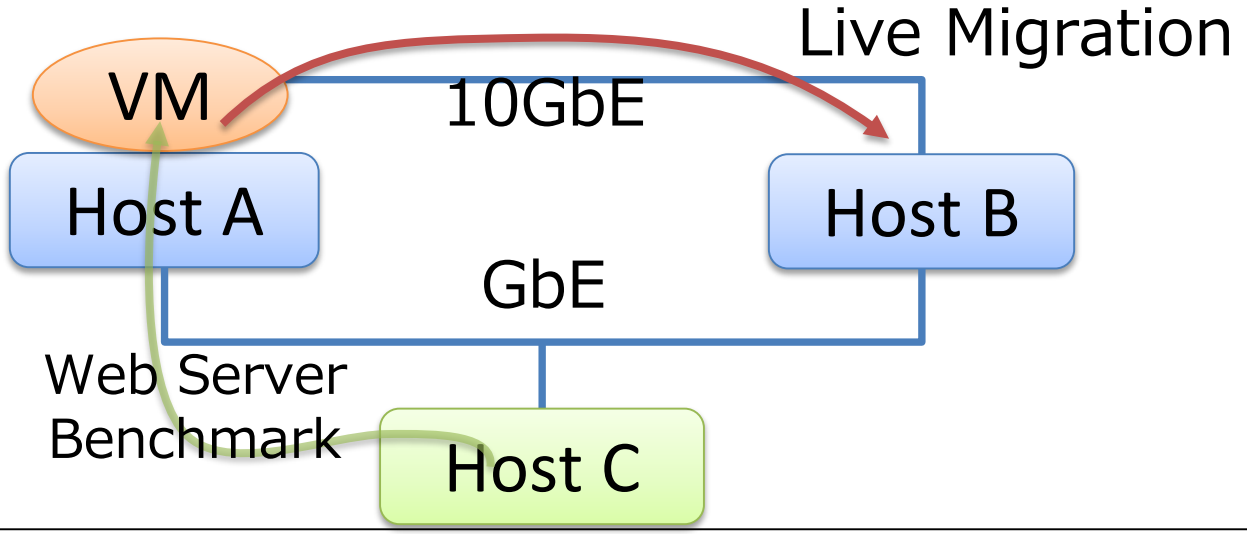
- KVMおよびLinuxが備えるAsynchronous Page Fault (APF)機能を拡張して実装
  - VMのメモリページがホストOS上でスワップアウトされていた時に、ページフォルト処理を非同期化する仕組み
  - APFに対応したKVMドライバおよびゲストOSカーネルであれば、そのまま提案機構にも対応
    - 実はAPFの割り込み通知をそのまま流用
  - 提案機構ではUMEMドライバおよびデーモンを非同期処理に拡張

評価の前に… 提案機構の注意点

- ゲストOS上で複数のCPU待ちプロセスが存在しないと効果がない
- ページ転送処理を非同期化できるのは、VCPUスレッドがゲストOS内を走行しているときに発生したページフォルトのみ
  - メインスレッドやディスクのI/Oスレッドは対象外
  - VCPUスレッドがゲストOS外でページにアクセスする処理（APICの処理等）は対象外
- 転送待ち中のページに対して、再度ページフォルトが発生すると、通常のページフォルト処理にフォールバックする
  - あるページの取得を複数のプロセスが待っている場合に発生

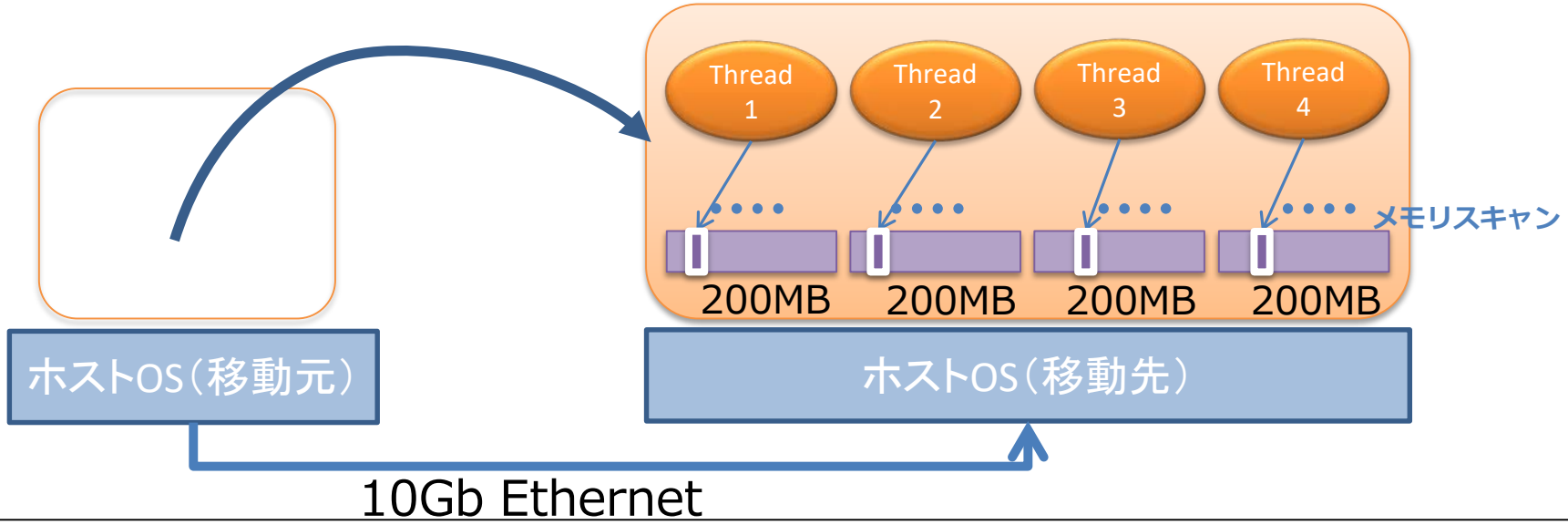
評価環境

- 物理ハードウェア
  - Intel Xeon E5620 x2 (EPTあり), 24GB RAM
  - マイグレーション用NIC : 10GbE (RTT 150us)
  - 負荷生成用NIC : GbE
- 仮想マシン
  - 1VCPU, 1GB RAM
- マイグレーション
  - メジャーフォルト発生ページの前後8ページを一度に転送
  - バックグラウンドの転送は無効



動作確認

- 単純なベンチ－マークを準備
  - 4スレッドがそれぞれ200MBのメモリ領域を確保
  - 実行ホスト切り替え後、各スレッドがそれぞれメモリ領域の先頭から末尾まで順番にページアクセス
  - メモリスキャンの完了時間を計測

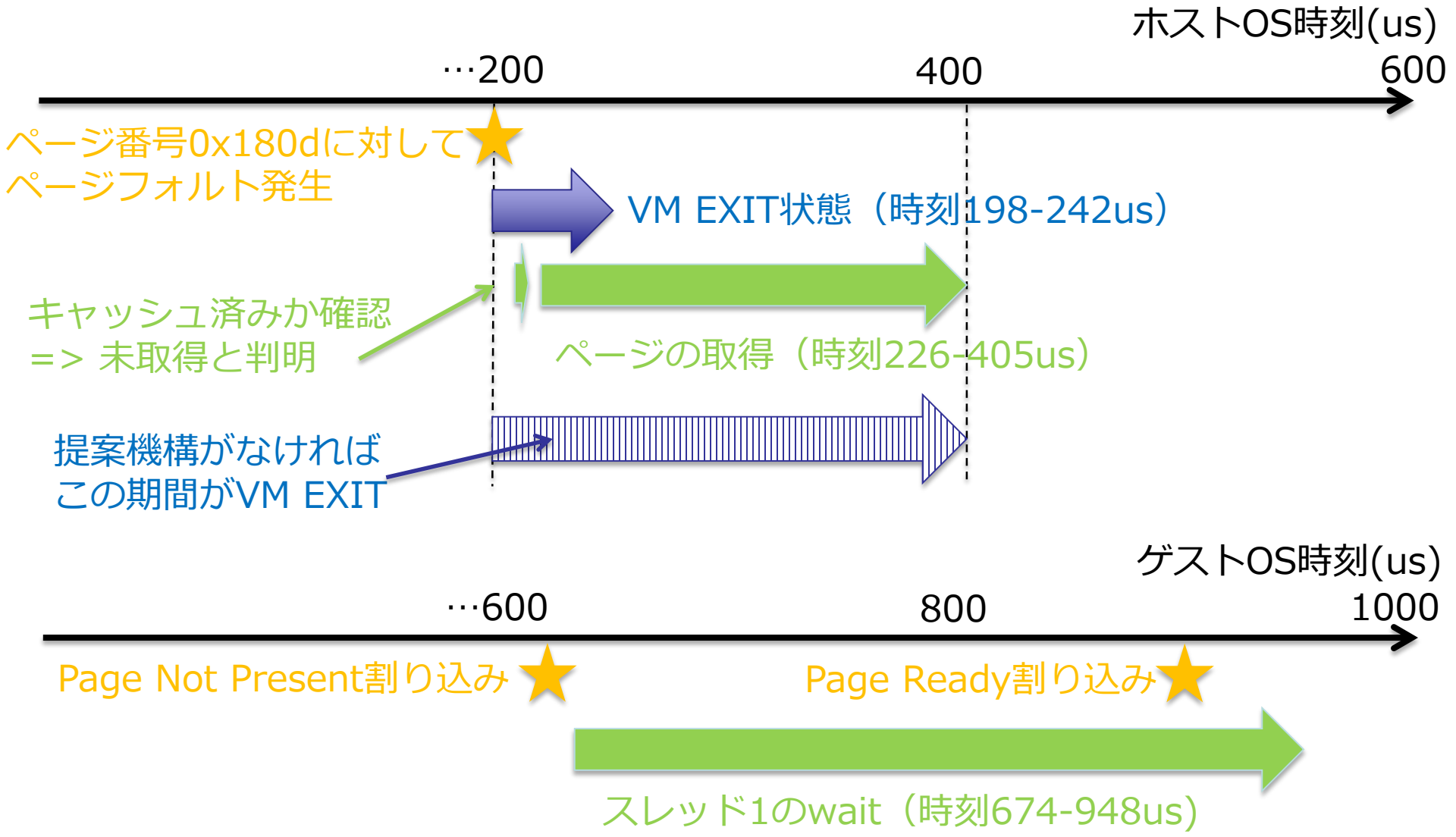


ベンチマーク結果

- 完了時間
  - 提案機構あり：9.8秒
  - 提案機構なし：17.6秒
- 単位実時間あたりのゲストOS実行時間
  - VMX Non Rootモードの走行時間
  - 提案機構あり：0.55
  - 提案機構なし：0.38

SystemTapによる提案機構の動作解析

スレッド1がページ番号0x180dに対してページフォルトを発生した場合

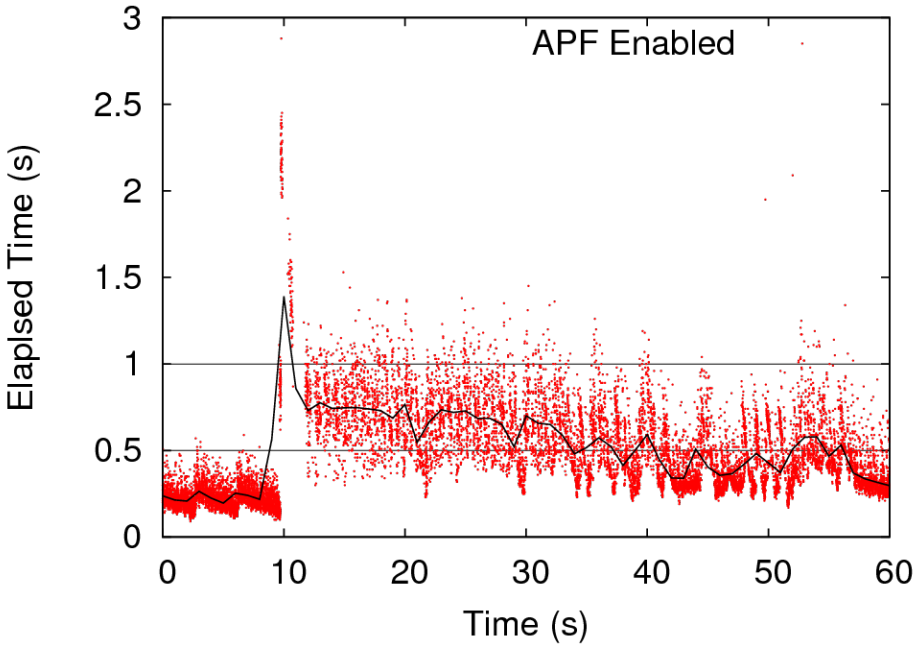


ウェブサーバを用いた評価

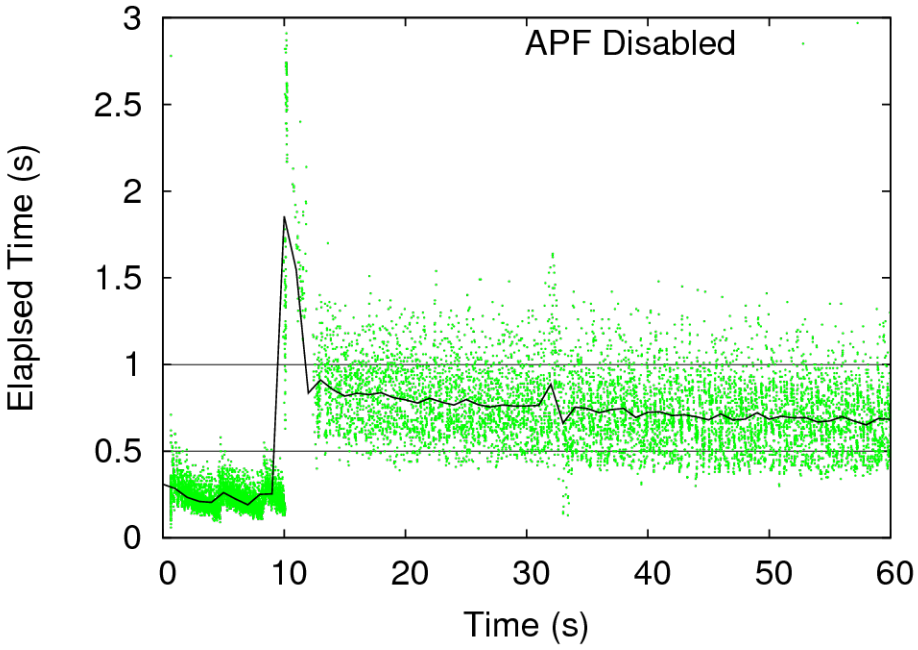
- VM
  - Apacheプレフォーク（最大128プロセス）
  - 静的ウェブコンテンツ 100KB x 5000個
    - 計測開始前にすべてページキャッシュに載せる
- クライアント
  - HTTPクライアントを128スレッド
  - 各スレッドはランダムな順番でウェブコンテンツにアクセス
  - 各リクエストの応答時間を計測

リクエスト応答時間

提案機構あり

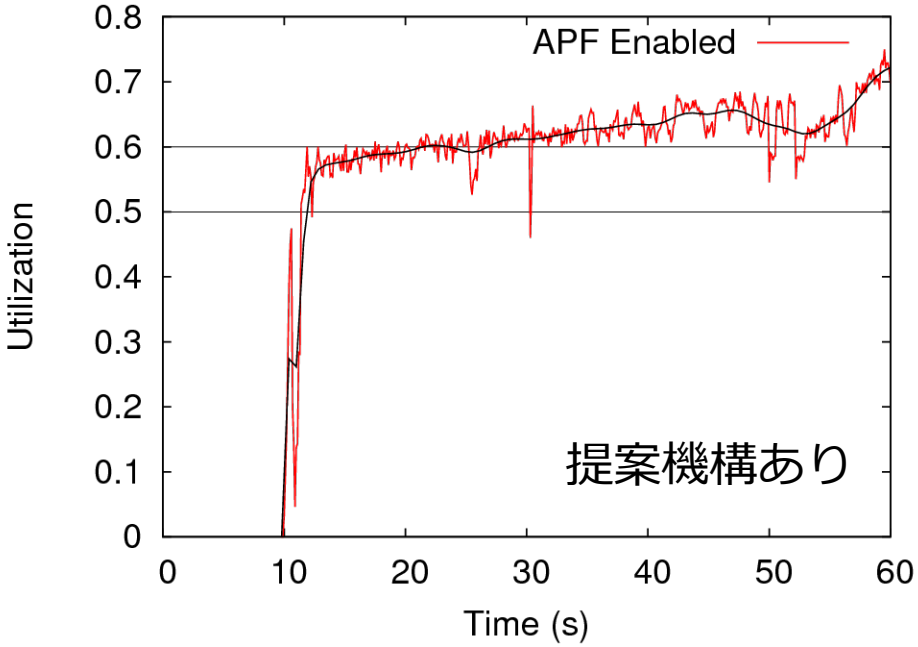


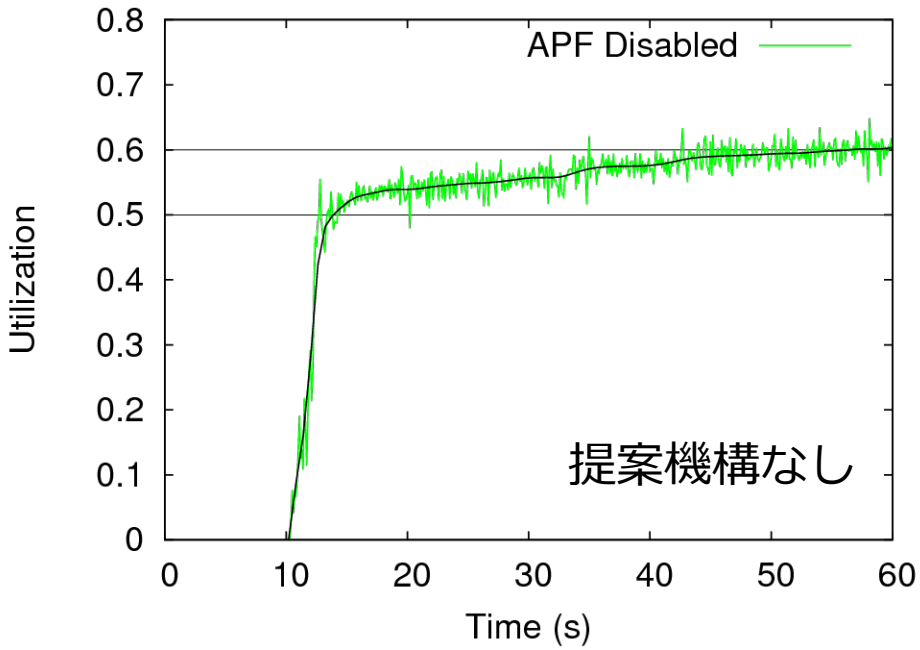
提案機構なし

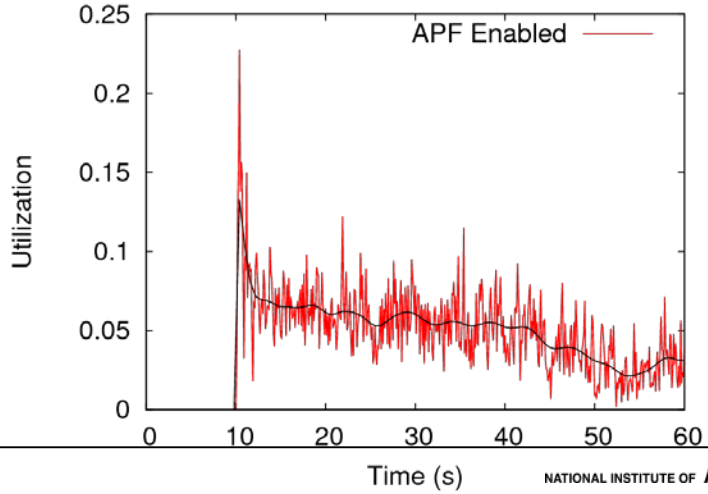


提案機構によりHTTPリクエストの応答時間の悪化を緩和できた。

単位実時間に占めるゲストOS実行時間の割合





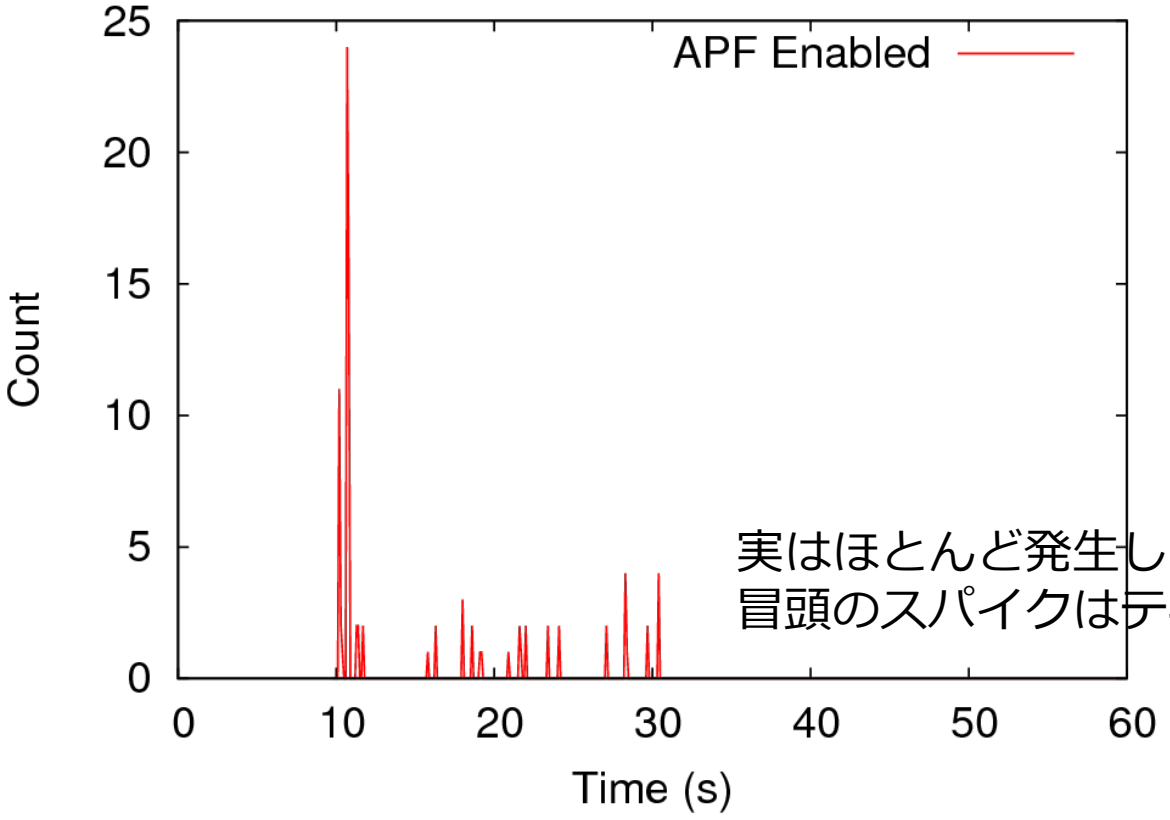


提案機構ありの場合において、
ページフォルト処理に並行して
ゲストOSを実行した時間

ゲストOS実行時間の差を概ね説明できる

転送待ち中のページに対して再度ページフォルトが発生した回数

100msあたりの発生回数



実はほとんど発生していない。
冒頭のスパイクはテキスト領域など。

関連研究（１）

- ダブルページング問題への対処手法
  - VMのメモリがホストOS上でスワップアウトされた時にVMのページフォルト処理遅延が増加
  - ページフォルト処理の非同期化
    - IBM z/VMのPseudo-Page-Fault Interruption
    - Linux KVMのAPF
  - 提案機構は同様の仕組みをポストコピー型ライブマイグレーションにはじめて適用

関連研究（２）

- ポストコピー型の性能向上手法
  - SnowFlockとその後継の研究
    - ポストコピー型によるVMの高速クローン機構
    - malloc()を準仮想化
      - 移動先でmallocしたページは転送しない
    - ゲストページテーブルの解析
      - 関連性が高いページ群をまとめてプレキャッシュ
  - Lagar-Cavillaらのポストコピー型
    - ページフォルト発生前後のページからプレキャッシュ
    - 我々のポストコピー型でも実装済み
  - プレコピー型とポストコピー型のハイブリッド
    - プレコピー型を開始し、その後ポストコピー型へ移行
    - ポストコピー時のキャッシュミスを低減
    - 我々のポストコピー型でも実装済み（要評価）
  - RDMAの利用
    - メジャーフォルトの処理遅延を低減

Yabusame開発史（１）

- 2009年度はじめ頃
  - 最初のプロトタイプを作り始める
  - それ以前はディスクのポストコピー型を作っていた
- 2009年8月 SWoPP
  - Qemu/KVMに対するVM本体のポストコピー型を研究発表
  - その後、IC2009、ComSys09、CCGrid10等で発表
  - その後、応用事例の研究もいくつか発表した
- 2010年半ば頃
  - プロダクションレベルの実装を開発する話が持ち上がる

Yabusame開発史（２）

- 2011年初頭
  - 再実装を開始（VA Linuxに依頼）
- 2011年夏頃
  - 対外的に発表
    - KVM Forum 2011
    - Linux Plumbers 2011
- 2012年初頭
  - Yabusameのコードを公開
  - APFを実装
- 2012年秋
  - KVM Forum 2012
  - 実装を再度ブラッシュアップ
  - プレコピーとポストコピーのハイブリッドを実装

開発与太話

- Qemu/KVMにマージしたいがなかなか難しい
  - プレコピーの改良をまずはできる限り行うべき
  - もっとエンタープライズな環境で評価しないとダメ
    - 本当にそれが必要っていう理由付けが必要
    - IBMやRedhatのビジネス戦略にのせる必要
- とはいえ、パッチを当てればいつでも使える状態になっているので、まずは是非使ってみてください！
  - どんな場合もAISTでサポートします。
  - いろんな環境で評価して頂けるとうれしいです。
  - 使って論文を書いて頂けるのもうれしいです。

今後の課題

- エンタープライズ環境への対応
  - 10GbEやInfiniband
  - RDMA化
  - など

- 産総研ではインターンを募集しています。
  - 上記ネタやその他Yabusameに関連する研究に取り組める方。
  - あなたの卒論、修論、博論ネタとなるように、開発だけではなく研究としてサポートします。
  - 結構な(?)バイト代払います。

まとめ

- ポストコピー型ライブマイグレーションの性能低下を緩和する手法を提案
  - VMが未転送のメモリページにアクセスしたら、ゲストOS全体を止めるのではなく、そのページにアクセスしたゲストOS上のプロセスのみ止め、他のプロセスの実行は継続する仕組み
  - ページフォルトを準仮想化することで実現
  - 既存のQemu/KVMと親和性の高い実装
    - Linux 2.6.38以降のカーネルであれば、改変を施すことなくゲストOSとして提案機構に対応
- 評価の結果、性能低下の緩和が確認できた。
- みんな遊んでみてね。**(qemu) migrate –p …**