McKernel概要

高木 将通
理研 計算科学研究機構

2015/12/14

アウトライン

- 背景
- LinuxとLight-Weight Kernelのハイブリッド手法
- Interface for Heterogeneous Kernels (IHK)
- McKernel
- HPC向けOSプロジェクト比較
- 文献リスト

背景 (1/3)

- HPCシステムの複雑化
  - 並列性の増大
  - メモリ階層数の増大
  - 電力制約

➡ スケーラブルで安定した性能の提供と
新ハードウェアへの迅速な追従が必要
（従来はAPIを絞ったLight-Weight Kernelで対応）

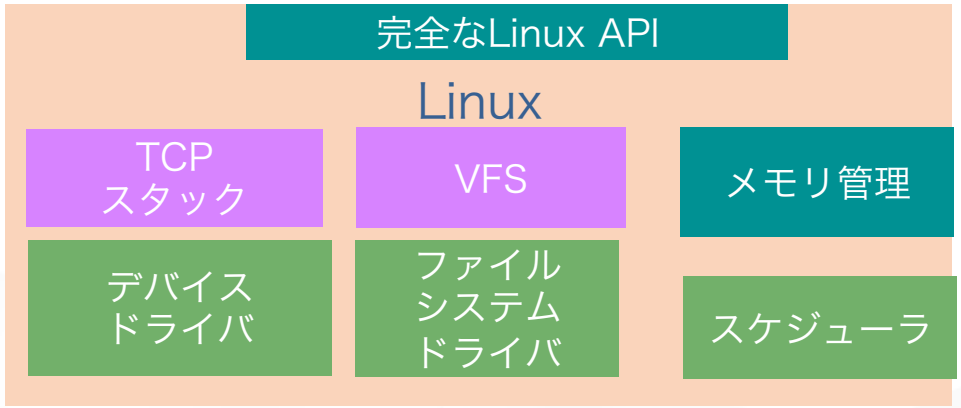
- アプリケーションの複雑化
  - 新しいプログラミングモデルの導入
    - 例：In-situデータアナリティクス、
      ワークフロー
  - 周辺ソフトの複雑化
    - 例：プロファイラ、ジョブスケジューラ、
      監視ソフト

➡ Linux APIを用いて開発されること
が多いため、Linux APIの提供が必要

➡ これら相矛盾する要件を満たせるか？

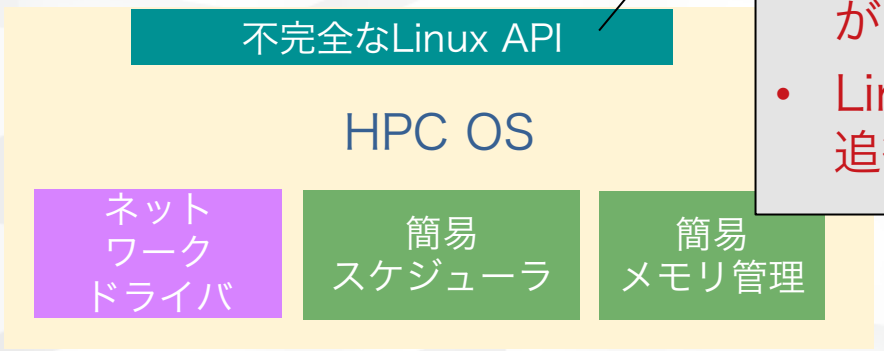
背景 (2/3)

従来のアプローチ１



LinuxからHPC性能を阻害する要因を除去

- OSノイズの除去
  - 例：デーモン、タイマ割り込み
- メモリ管理の簡略化
- スケジューラの簡略化



- 阻害要因除去を徹底すると完全なLinux APIサポートができなくなる
- Linuxカーネル変更への追従が困難

“Stripped down Linux” アプローチ

- 例：ZeptoOS, Cray’s Extr. Scale Linux, Fujitsu’s Linux,

背景 (3/3)

従来のアプローチ２

- スクラッチから作成した軽量カーネル (Light-Weight Kernel, LWK)から始める
- スケーラビリティを損なわないように機能を追加してLinux APIに近づける
  - 例
    - システムコール
    - /procファイルシステム
    - ダイナミックリンクライブラリ
    - スレッドのコアへのオーバーサブスクリプション

- 完全なLinux APIサポートができなくなる
- デバイスドライバの新規開発が必要

“Enhanced LWK” アプローチ
例：Catamount, CNK, Kitten

アプローチ

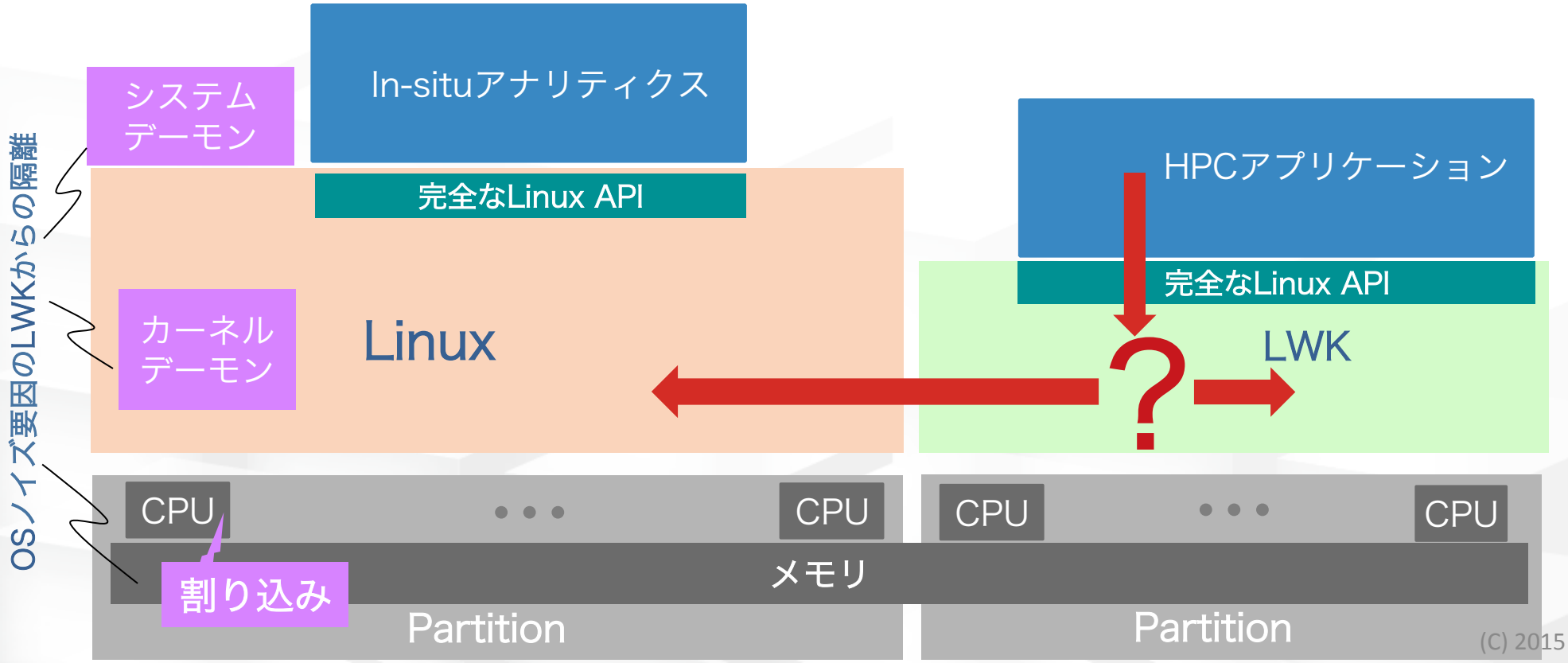
LinuxとLWKのハイブリッド

- 資源を２つのパーティションに分けLinuxとLWKを並列動作、
  またHPCアプリをLWK上で動作

→性能安定性・スケーラビリティ・新ハードウェアへの迅速な追従

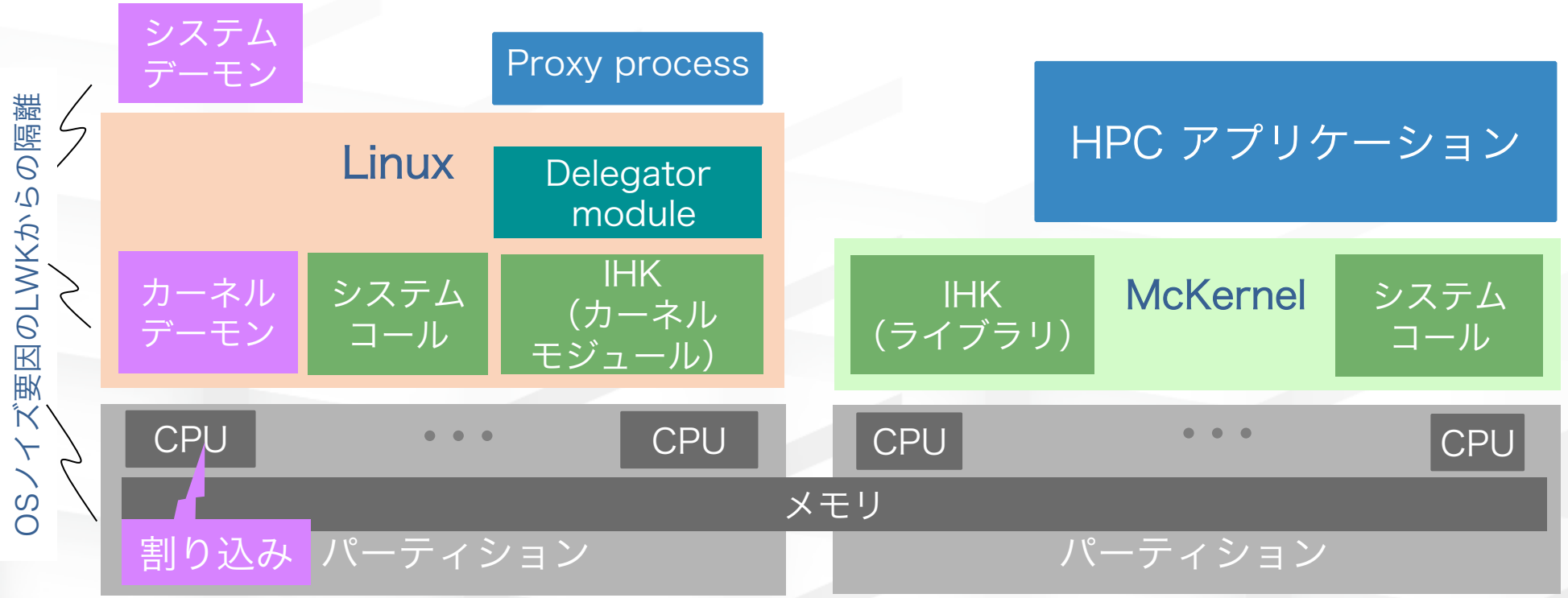
- OSサービスの一部をLWKで実行、他をLinuxにオフロード

→Linux APIサポート



アーキテクチャ

1. Interface for Heterogeneous Kernels (IHK): ハイブリッドOSのフレームワーク
   - ノード資源のパーティショニング
   - LWKの管理（例：カーネルの起動、停止)
   - LWKとLinuxの間の通信機構（Inter-kernel communication, IKC)の提供
2. McKernel: スクラッチから開発されたHPC向けLWK
   - ノイズレス
   - 性能クリティカルなシステムコールのみMcKernelで動作、他はLinuxにオフロード



IHKのオーバービュー

目的

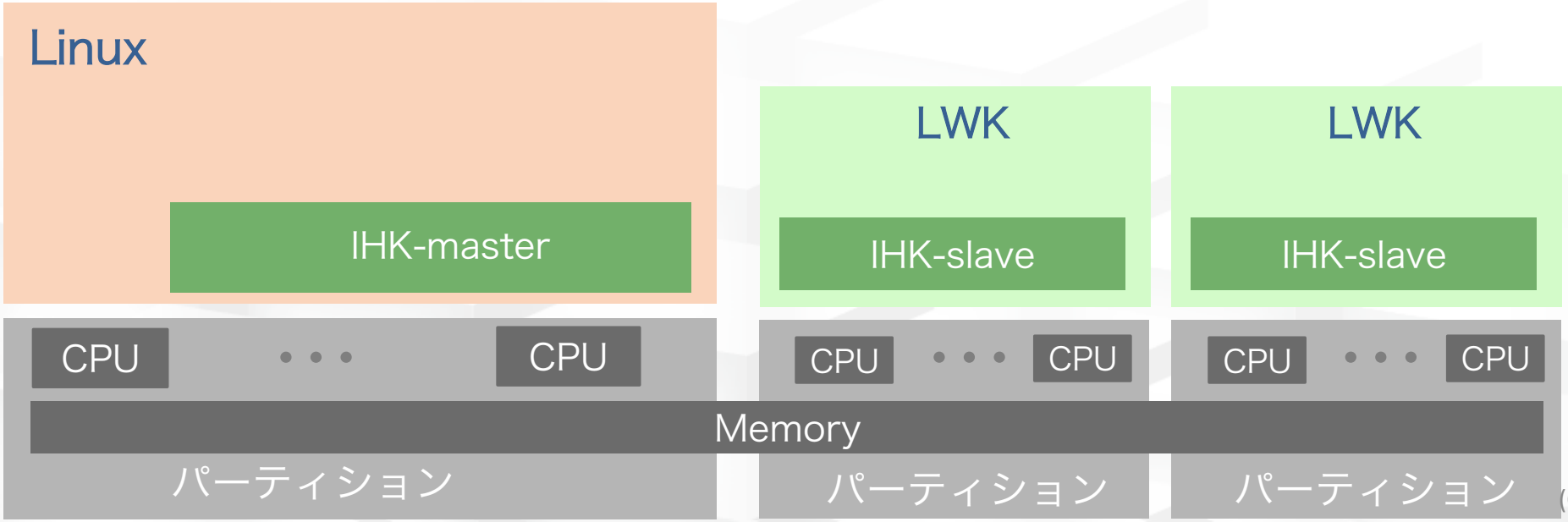
- OSカーネルの新プロセッサアーキ対応プロトタイピングの容易化

構成

- LinuxのカーネルモジュールであるIHK-masterとLWKのライブラリであるIHK-slave
  - Linuxのソースコードの変更不要

機能

- ノードの再起動なしのノード資源のパーティショニング
- LWKの起動停止
- IKCの提供



IHKの構造

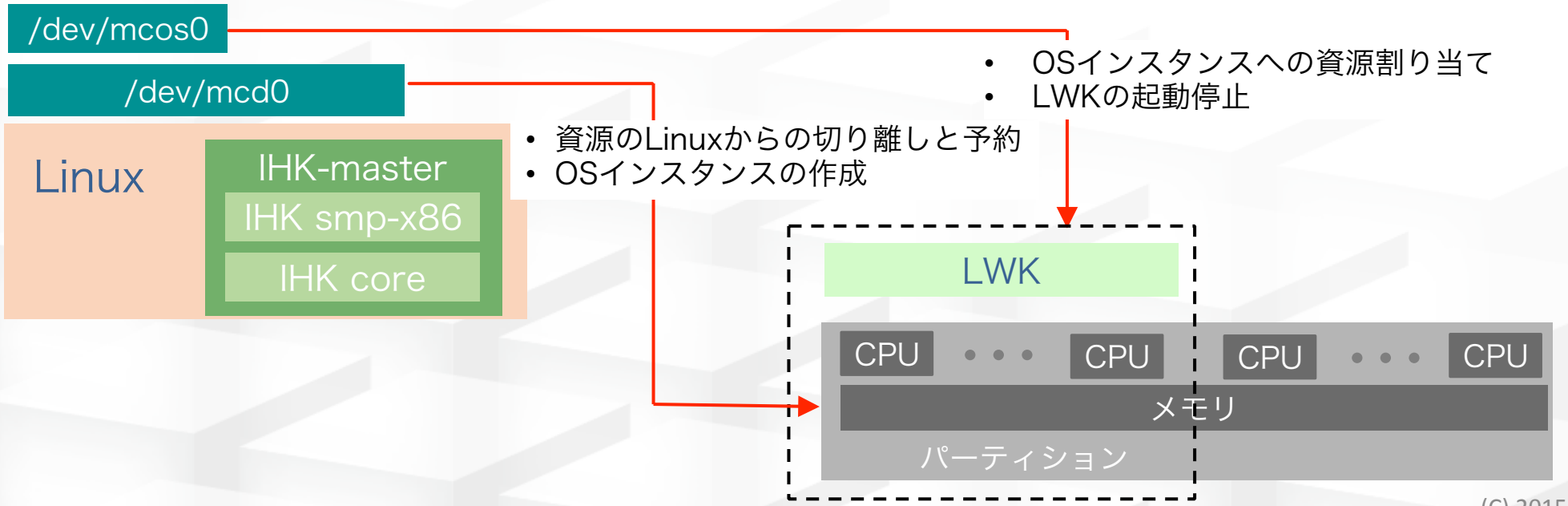
複数アーキや構成をサポートするため階層構造をとる

- 構成：スタンドアローンプロセッサのみを使う、ホストプロセッサ＋PCIバス接続アクセラレータ

1. アーキや構成に共通のモジュール（IHK core）
   - アーキや構成に依存するモジュールの登録
2. アーキや構成に依存するモジュール
   - IAアーキ、スタンドアローン構成向け（smp–x86）
   - IAアーキ、Intel XeonにPCIバス接続タイプのXeon Phiを接続する構成向け

管理ツール向けインターフェイス

- 資源の操作：/dev/mcdX
- OSインスタンスの操作：/dev/mcosX



McKernelのオーバービュー

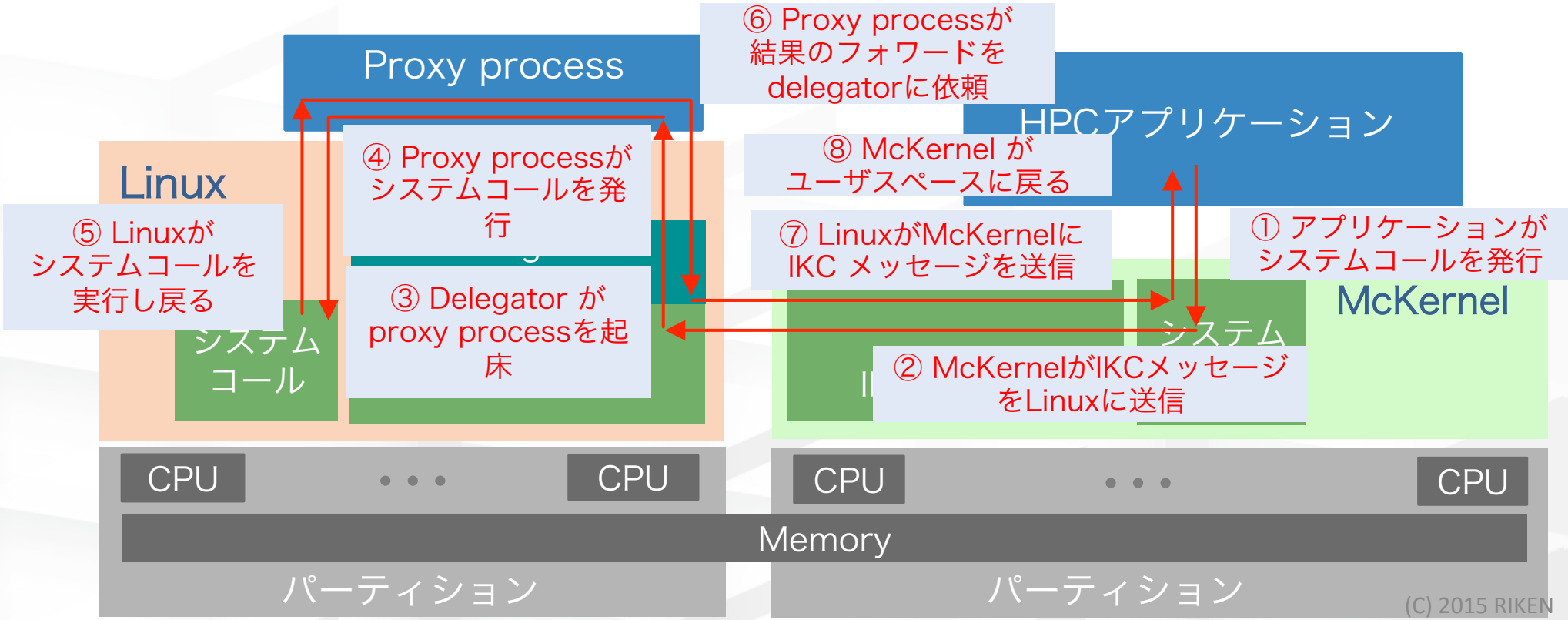
- HPC向けLWK
- ノイズレス
- Linux APIサポート
  - 性能クリティカルなシステムコールのみMcKernelで動作、他はLinuxにオフロード
  - Linux Test Projectのテストスイートの100%パス

分類	実装済み	実装予定
プロセス管理	arch_prctl, clone, execve, exit, exit_group, fork, futex, get_cpu_id, getpid, get{u,g}id, gete{u,g}id, getres{u,g}id, getppid, gettid, kill, pause, ptrace, rt_sigaction, rt_sigpending, rt_sigprocmask, rt_sigqueueinfo, rt_sigreturn, rt_sigsuspend, set_tid_address, set{u,g}id, setre{u,g}id, setres{u,g}id, setfs{u,g}id, setpgid, sigaltstack, tgkill, vfork, wait4, waittid	{get,set}rlimit, {get,set}_thread_area, rt_sigtimedwait, signalfd, signalfd4
メモリ管理	brk, madvise, mincore, mlock, mmap, mprotect, mremap, msync, munlock, munmap, process_vm_{readv,writev}, remap_file_pages, set_robust_list, shmat, shmctl, shmdt, shmget	{get,set}_mempolicy, {get,set}_robust_list, mbind, migrate_pages, mlockall, modify_ldt, move_pages, munlockall
スケジュール	getcpu, gettimeofday, nanosleep, sched_getaffinity, sched_setaffinity, sched_yield	alarm, {get,set}itimer, settimeofday, time, times
パフォーマンスカウンタ	Original functions: pmc_init, pmc_start, pmc_stop, pmc_reset	PAPI functions

- 上記以外のシステムコールはLinuxに移譲

McKernelの機能ーシステムコール移譲

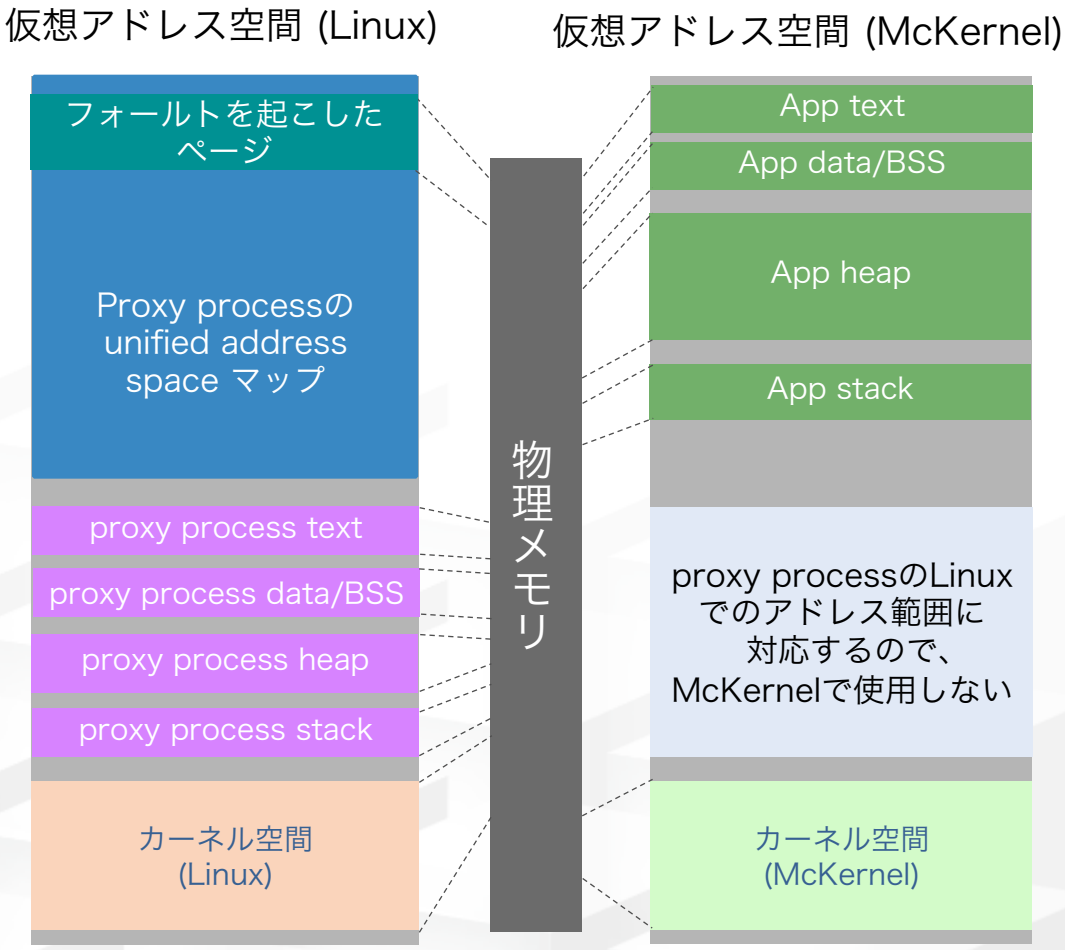
- McKernel上で動作するプロセスには対応するLinux上で動作するProxy processが存在
  - Linux側でシステムコールの引数を受け取り、McKernelプロセスの代わりにシステムコールを実行する
  - システムコール発行時にMcKernelプロセスかのように振る舞うために、McKernelのプロセスから見た実行コンテキストをProxy processから参照可能にする
    1. 仮想アドレス空間をMcKernelのプロセスと共有する
    2. プロセスID、ファイルディスクリプタテーブルはProxy process側で記録管理



McKernelの機能－Unified Address Space

- システムコールのポインタ引数変換オーバーヘッドを削減
  - 例：read()システムコールのターゲットバッファ
- McKernelプロセスと同じ仮想物理マップをProxy processに持たせることで実現

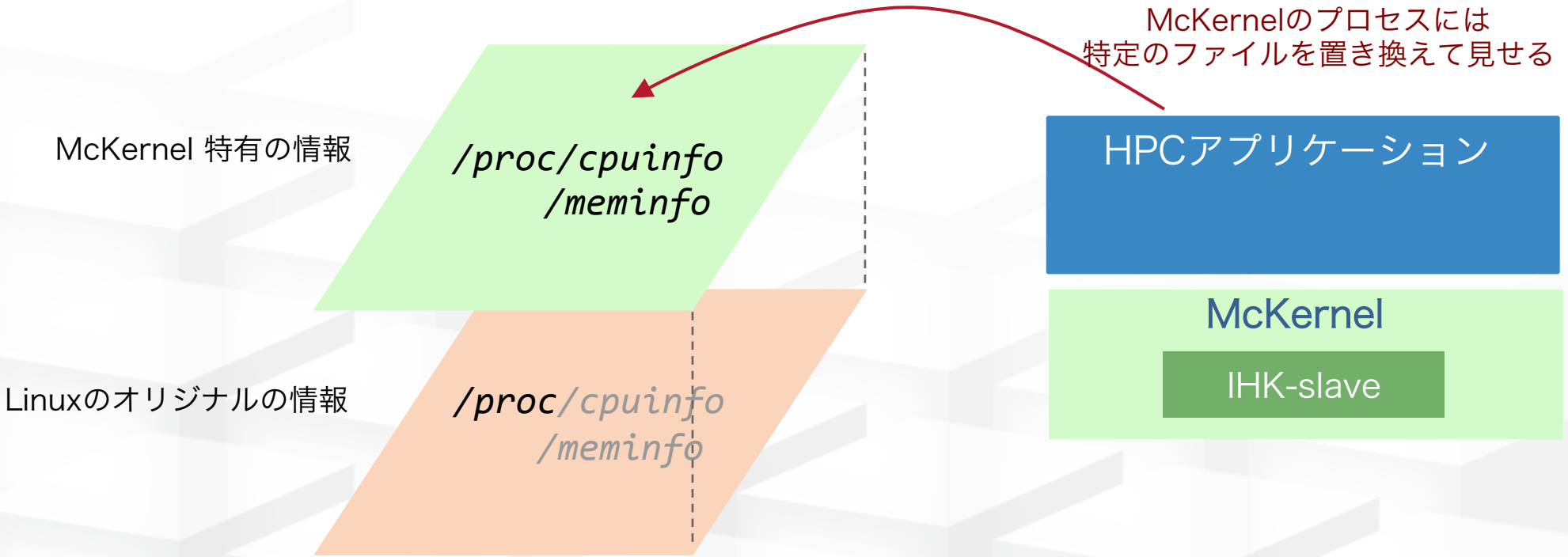
1. Proxy processのtext、data、bssセグメントをMcKernelプロセスが使用しないアドレスに移動して起動
2. Linux側でMcKernelプロセスのページテーブルアドレスを記録、またページフォールトをフック
3. Linux側のシステムコール処理でポインタ引数でページフォールトが起きたら、McKernelプロセスのPTEを参照
   - エントリがない場合にはMcKernelにページフォールトを依頼
4. Proxy processの同じ仮想アドレスに物理ページをマップ



McKernelの機能－/procと/sysファイルシステム (1/2)

- 要件
  - McKernel上で動作するプロセスにMcKernelのシステムまたはプロセスの情報を/procまたは/sysのパスで見せる

  ＝ McKernelのプロセスに対し、特定の/procまたは/sysのファイルについて、McKernel特有の情報で置き換えて見せる

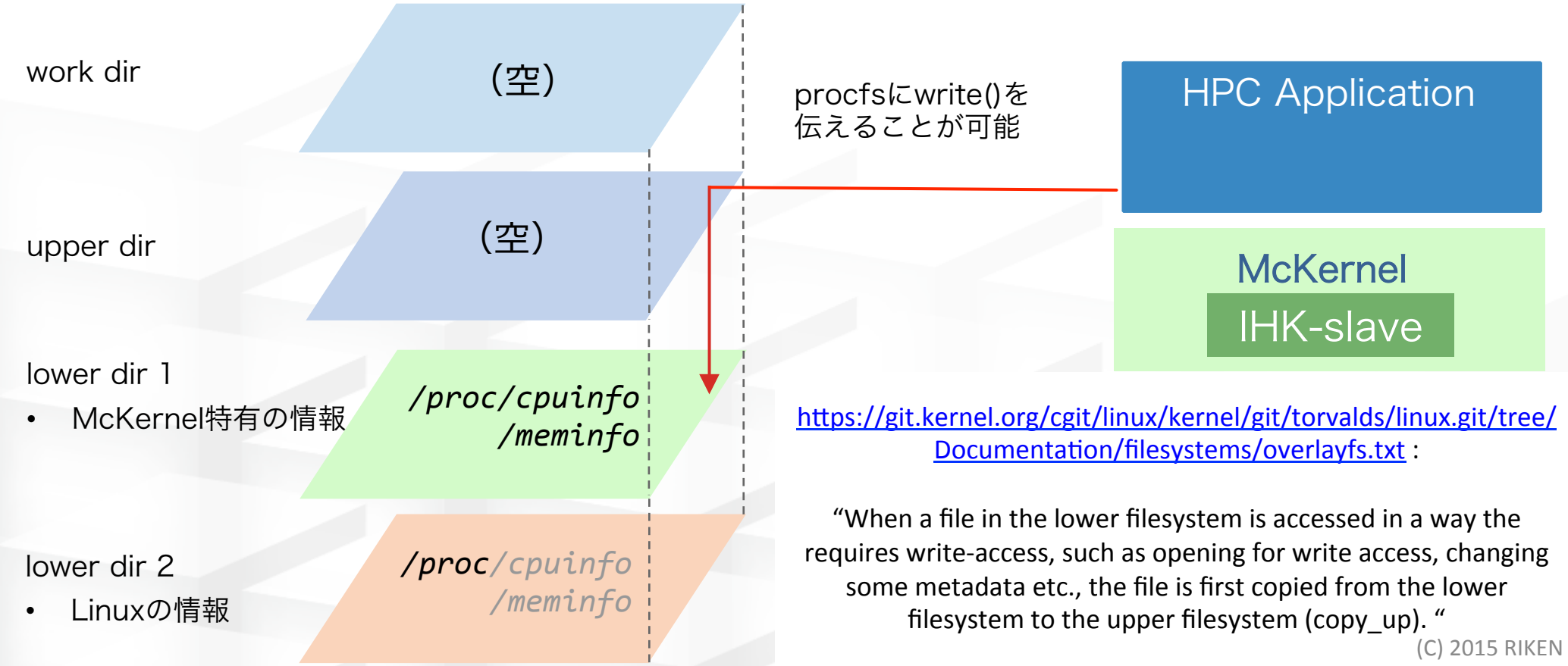


McKernelの機能ー/procと/sysファイルシステム (2/2)

- 複数ディレクトリを優先度付きで重ねることのできるOverlayfsを改造して実装
  - upper dirはファイル作成が可能でなければならないが、procfsはファイルの作成方法が特殊

  → lower dir同士を重ねる
  - Overlayfsはwrite()システムコールでファイルの変更を行おうとすると、コンテンツを"upper"ディレクトリにコピーしてしまいprocfsにwrite()が伝達されない

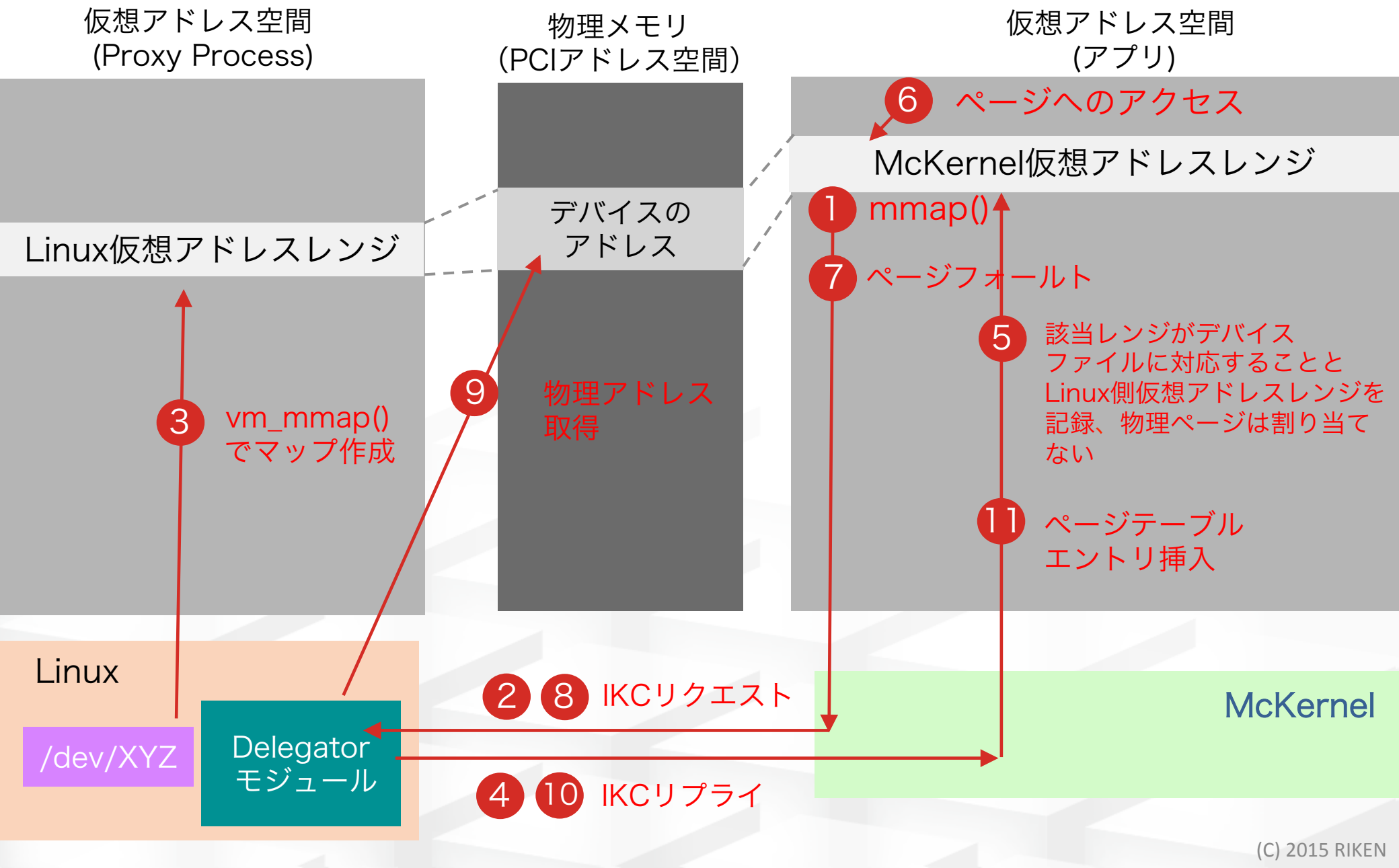
  → これを行わないオプションを追加



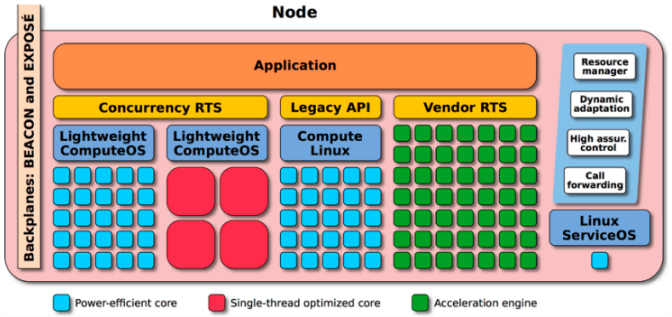
https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/filesystems/overlayfs.txt :

"When a file in the lower filesystem is accessed in a way the requires write-access, such as opening for write access, changing some metadata etc., the file is first copied from the lower filesystem to the upper filesystem (copy_up). "

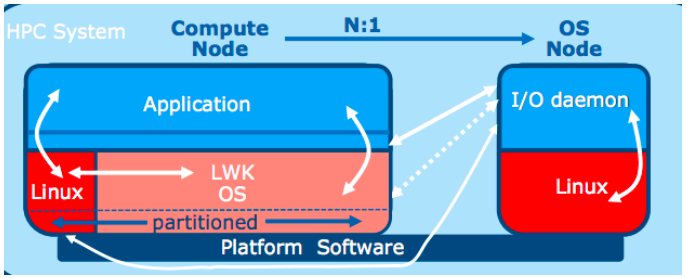
McKernelの機能ーデバイスファイルマップ



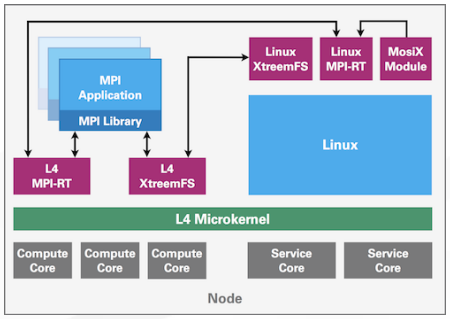
HPC向けOS比較



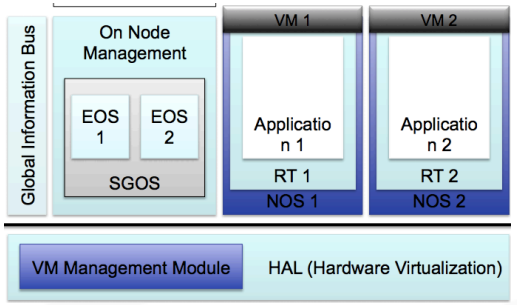
Argo (Argonne National Laboratory)



mOS (Intel)



SPPEXA (TU Dresden)



Hobbes (Sandia National Laboratories)

プロジェクト	Linuxカーネル変更要否	Linux用デバイスドライバの軽量カーネルからの利用	カーネルレベルでの隔離	Linux APIの完全なサポート	開発難易度
Argo	必要	Yes	No	Yes	小
mOS	必要	Yes	No	Yes	小
Hobbes	不要	No	Yes	No	大
SPPEXA (L4+Linux)	?	?	Yes	No?	大?
IHK/McKernel	不要	Yes	Yes	Yes	大

論文リスト

- Balazs Gerofi, Takagi Masamichi and Yutaka Ishikawa: "Toward Operating System Support for Scalable Multithreaded Message Passing", In Proc. EuroMPI, 2015
- Balazs Gerofi, Masamichi Takagi, Yutaka Ishikawa, Rolf Riesen, Evan Powers and Robert W. Wisniewski: "Exploring the Design Space of Combining Linux with Lightweight Kernels for Extreme Scale Computing", In Proc. ROSS, 2015
- Masamichi Takagi, Balazs Gerofi, Norio Yamaguchi, Takahiro Ogura, Toyohisa Kameyama, Atsushi Hori, Yutaka Ishikawa: "Operating System Design for Next Generation Many-core based Supercomputers", IPSJ SIG Technical Reports, 2015-ARC-215(1), pp. 1–8, 2015
- Balazs Gerofi, Akio Shimada, Atsushi Hori, Takagi Masamichi and Yutaka Ishikawa: "CMCP: A Novel Page Replacement Policy for System Level Hierarchical Memory Management on Many-cores", In Proc. HPDC, 2014
- Taku Shimosawa, Balazs Gerofi, Masamichi Takagi, Gou Nakamura, Tomoki Shirasawa, Yuji Saeki, Masaaki Shimizu, Atsushi Hori and Yutaka Ishikawa "Interface for Heterogeneous Kernels: A Framework to Enable Hybrid OS Designs targeting High Performance Computing on Manycore Architectures", In Proc. HiPC, 2014
- Balazs Gerofi, Akio Shimada, Atsushi Hori and Yutaka Ishikawa: "Partially Separated Page Tables for Efficient Operating System Assisted Hierarchical Memory Management on Heterogeneous Architectures", In Proc. CCGRID, 2013
- 佐伯 裕治, 清水 正明, 白沢 智輝, 中村 豪, 高木 将通, Balazs Gerofi, 思 敏, 石川 裕, 堀 敦史, 「ヘテロジニアス計算機上のOS機能委譲機構」, 情報処理学会, 2013-OS-125(15) , pp. 1-7, 2013

Q&A

- ご静聴ありがとうございました
- ご質問がございましたらお願いいたします