次世代PCクラスタのための システムソフトウェア

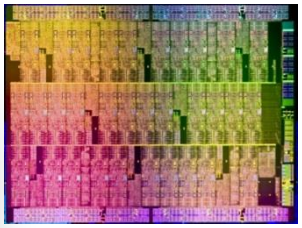
14:20 – 14:45

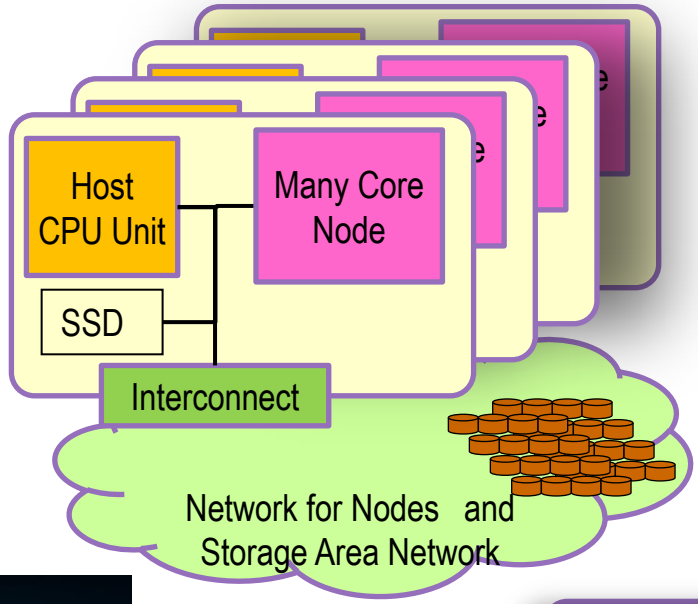
システムソフトウェア技術部会
石川裕（東京大学）

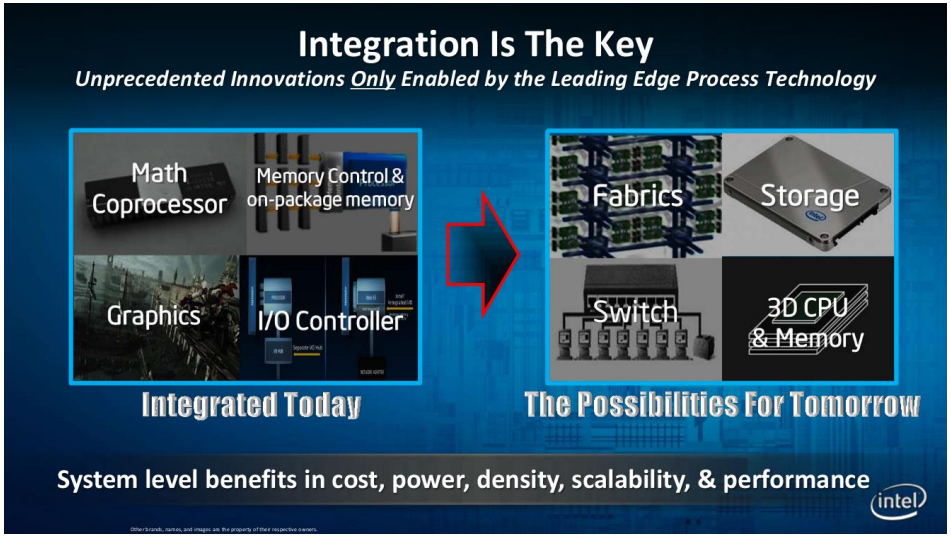
謝辞：IHKおよびMcKernelの開発の一部は文部科学省「将来のHPCIシステムのあり方の調査研究」における課題名「レイテンシコアの高度化・高効率化による将来のHPCIシステムに関する調査研究」の支援を受けている

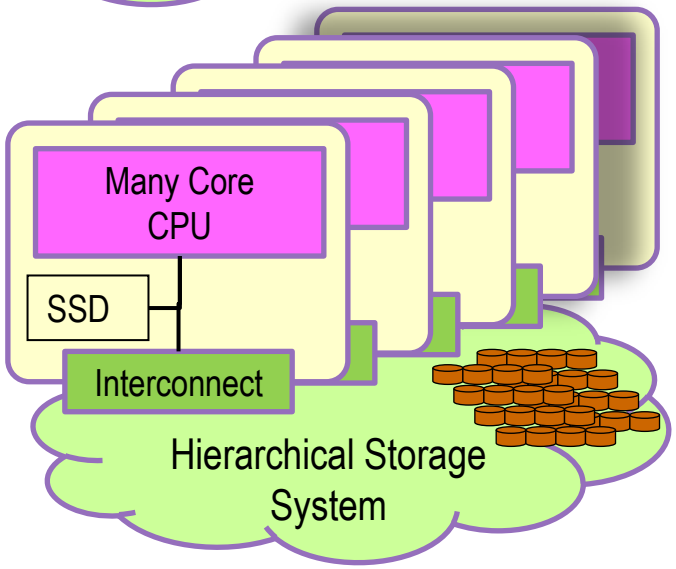
今後のPCクラスタ像

- HPC向けコモディティ製品の動向
  - メニーコア化
  - メモリ階層
    - キャッシュ、オンパッケージ、オンボード、SSD、HDD
  - インターコネクト









出展：ISC'13 Keynote by Raj Hazra, Intel

なぜ新しいOS？

A Novel OS architecture is required to support the following features:

Core Capability	O(100) Core / Node	O(1M+) Node, O(100M+) Core
• Reducing cache pollution • Localizing data • Managing memory hierarchy	• Reducing memory contention • Reducing data movement among cores • Providing fast communication • Parallelizing OS functions achieving less data movement	• Providing scalable process management, communication, and file I/O • Providing fault resilience

- Exploring such a novel operating system is huge design space
  - New OS mechanisms and APIs are revolutionarily/evolutionally created and examined, and selected
- Needs a research vehicle, also deployable
  - To enable kernel developers quickly to program a new kernel feature and evaluate it
  - To share ideas and their implementations

なぜ新しいOS？

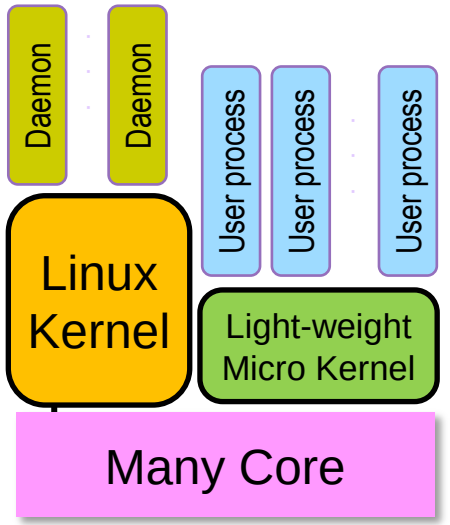
- 先端アーキテクチャ上のシステムソフトウェアの未成熟
  - 我々が必要とするシステムソフトウェア環境がタイムリに提供されるか？
  - 要求する基本性能・スケーラビリティを満たすことができるのか？
- Linuxカーネルのメニィコア対応 vs. 新OS
  - メニィコア向けの独自拡張を維持する手間 vs. 新OSの維持コスト
- ユーザに対してはLinuxカーネルAPIが提供できれば良い
  - LinuxカーネルAPI維持コストがどのくらいか？

Lunux 3.12.2の行数（一部）

init	3,482
kernel	204,523
mm	98,771
ipc	9,086
fs (core)	65,048
arch/x86	300,301
include	311,707
TOTAL	992,918

開発中カーネル行数

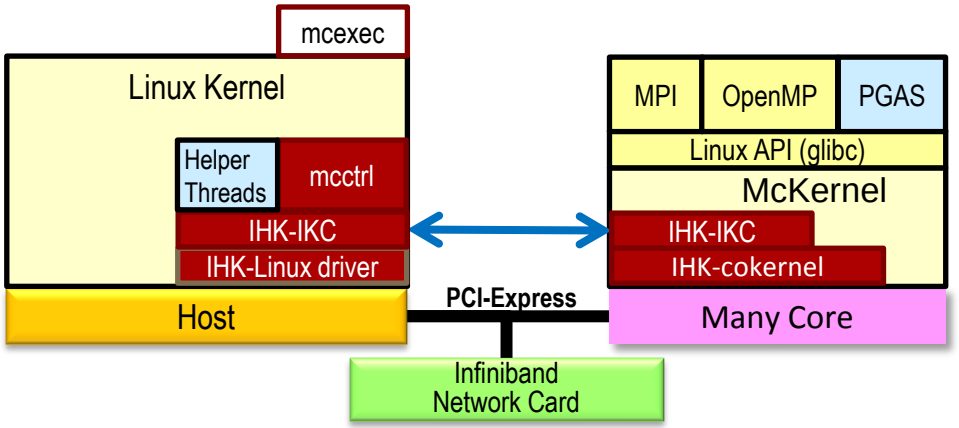
mckernel	28,339
ihk	23,608
TOTAL	51,947

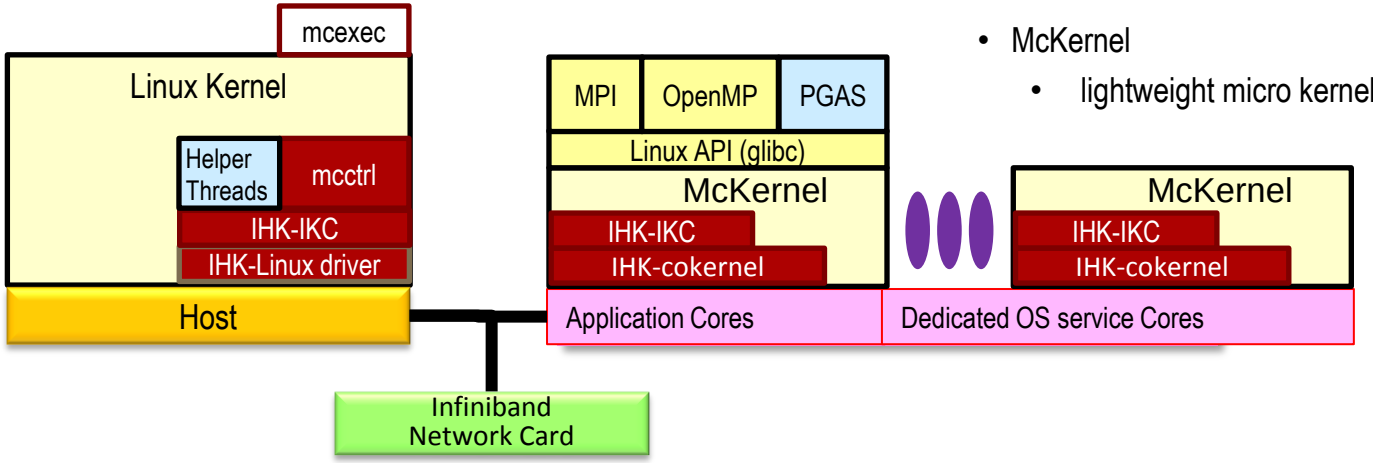


IHK and McKernel in Xeon Phi

In case of Non-Bootable Many Core

(This configuration is called “Attached”)





Features implemented so far in KNC

- glibc and pthread
  - Thread and memory management
  - File I/O, delegated to Linux in host
  - Memory map and dynamic link library
- Process launcher in host
- Direct Communication with InfiniBand
- MPI library (not fully) running on Xeon Phi
- OpenMP environment with Intel compiler

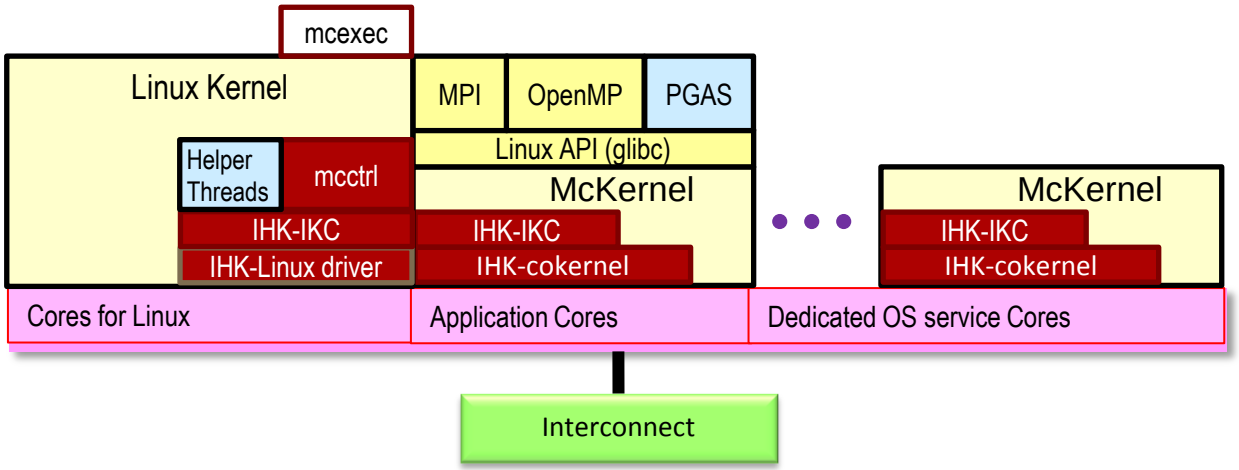
- IHK (Interface for Heterogeneous Kernel)
  - IHK-Linux driver: provides the functions of co-processors, such as booting, memory copy, and interrupt
  - IHK-cokernel: Abstracts the hardware functions of the manycore devices
  - IHK-IKC: providing communication between the Linux and co-kernel
- McKernel
  - lightweight micro kernel running on co-processors

IHK and McKernel

In case of Bootable Many Core

(This configuration is called “Built-in”)

IHK and McKernel also run on Xeon and the vmware virtual machine

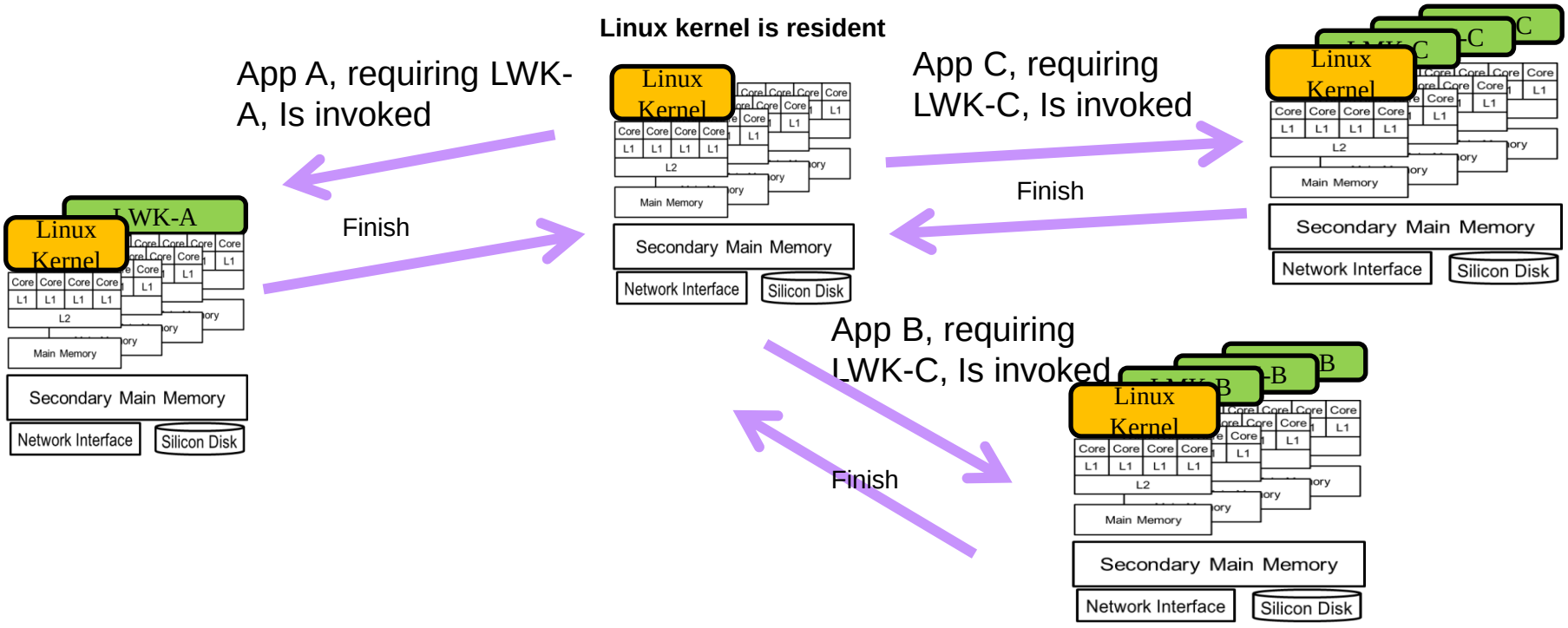


- IHK (Interface for Heterogeneous Kernel)
  - IHK-Linux driver: provides the functions of co-processors, such as booting, memory copy, and interrupt
  - IHK-cokernel: Abstracts the hardware functions of the manycore devices
  - IHK-IKC: providing communication between the Linux and co-kernel
- McKernel
  - lightweight micro kernel running on co-processors

Features implemented so far in KNC
- glibc and pthread
  - Thread and memory management
  - File I/O, delegated to Linux in host
  - Memory map and dynamic link library
- Process launcher in host
- Direct Communication with InfiniBand
- MPI library (not fully) running on Xeon Phi
- OpenMP environment with Intel compiler

計算ノード上のOSカーネル

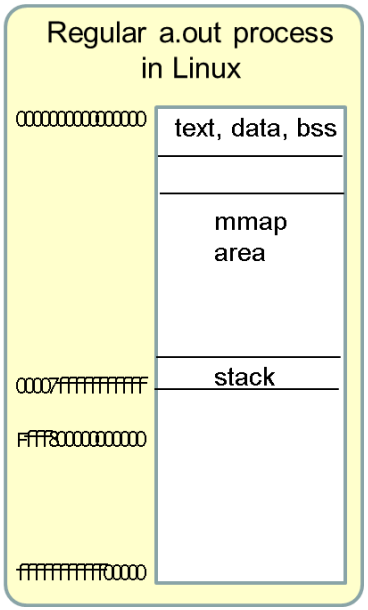
- Linux Kernel＋Loadable LWMK (McKernelとは限らず)
  - アプリケーションによってLWMKを入れ替えられるようにする
    - Linuxが動いているコア以外のコア全てを使うLWMK
    - Linuxが動いているコア以外のコアを分割していくつかのLMKが動作
  - LWMKのuserlandはLinux API踏襲

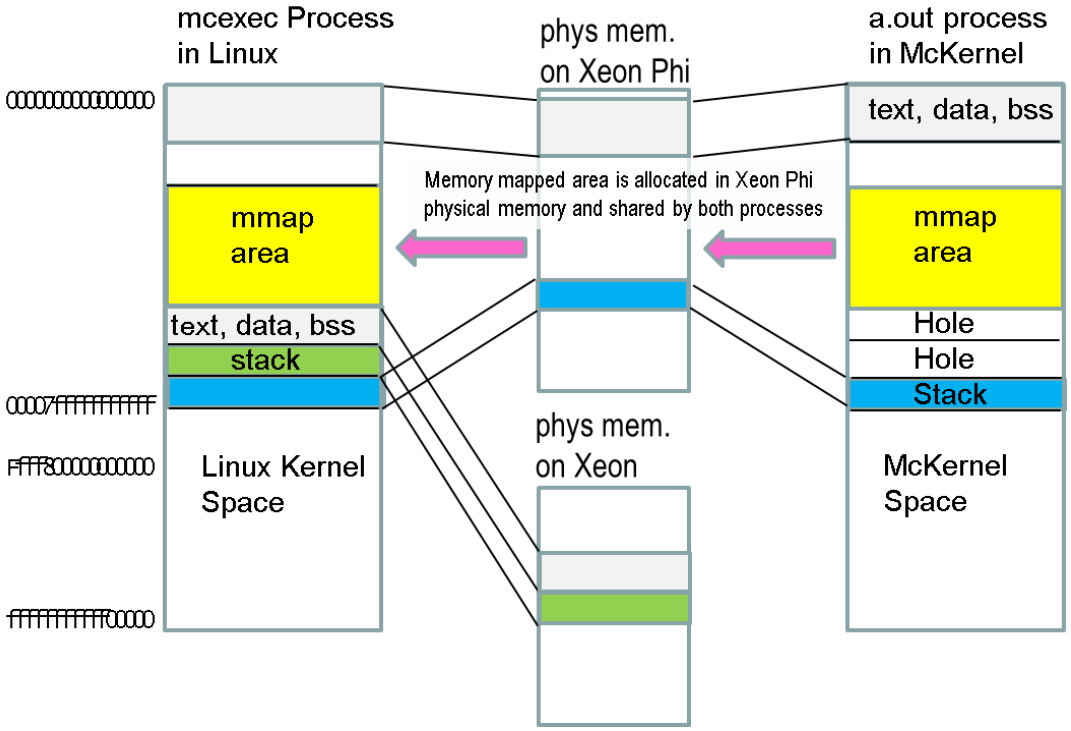


How an application program is executed (1/2)

$ mcexec ./a.out

- The mcexec program is compiled with options –fPIE –pie
- The mcexec binary is loaded in the different address than the regular a.out binary
- It creates an a.out process in McKernel
- The a.out user virtual memory space in McKernel is shared by the same address space in the mcexec process.





How an application program is executed (2/2)

$ mcexec ./a.out

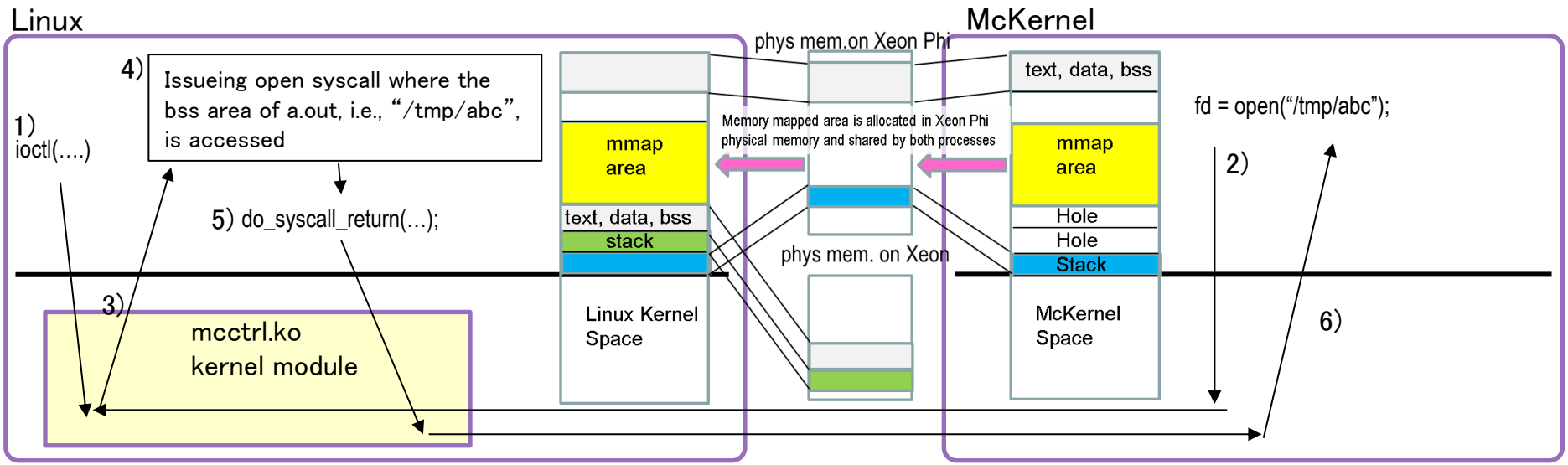
part of a.out code

```
01:  int fd; /* bss area */
02:  main() {
03:        int nr; char buf[512]; /* stack area */
04:        fd = open("/tmp/abc"); /* string is in data area */
05:        nr = read(fd, buf, 512);
06:        /* .... */ }
```

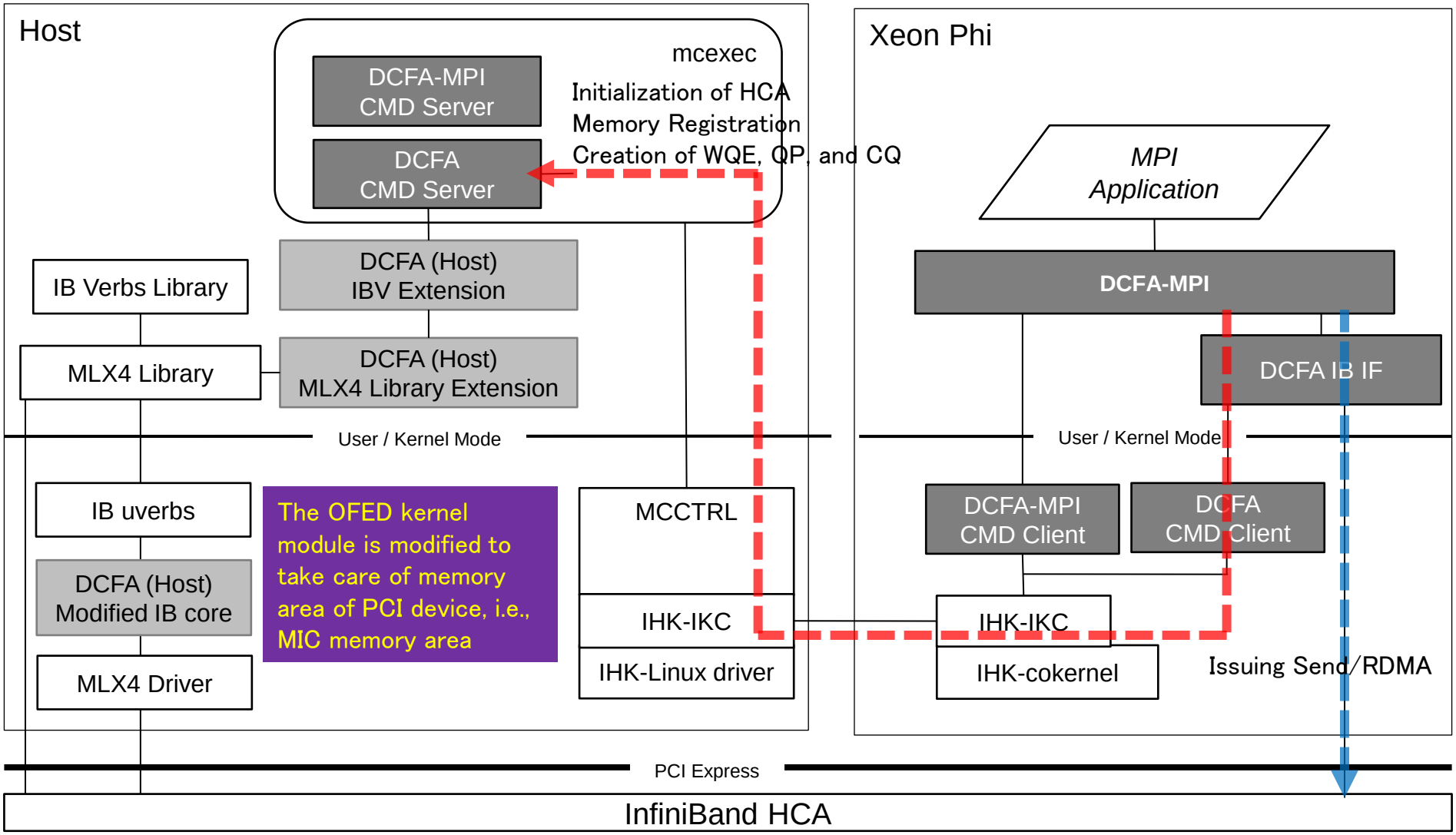
part of mcexec code

```
while (ioctl(fd, MCEXEC_UP_WAIT_SYSCALL, &w)
        == 0) {
    switch (w.sysnumber) {
    case SYS_OPEN: /***/; break;
    case SYS_EXIT:/***/; break;
    /* .... */
    default:   do_generic_syscall(…); }
    do_syscall_return(…); }
```

1) Linux: ioctl is issued to receive requests from McKernel
2) McKernel: The open system call is issued by a.out. McKernel forwards this request to Linux with arguments without any modification of argument addresses (not copy).
3) Linux: the request is received and returned to the mcexec
4) Linux: mcexec performs the request of the system call. In case of open system call, the file name is checked whether or not it is under /proc or /sys special directory. If so, special handling for MIC, others regular open system call is invoked.
5) Linux: mcexec issues do_syscall_return to send the result back to McKernel
6) McKernel: When the reply is received, the result is returned to the user program.



DCFA and MPICH/DCFA



McKernelシステムコール群

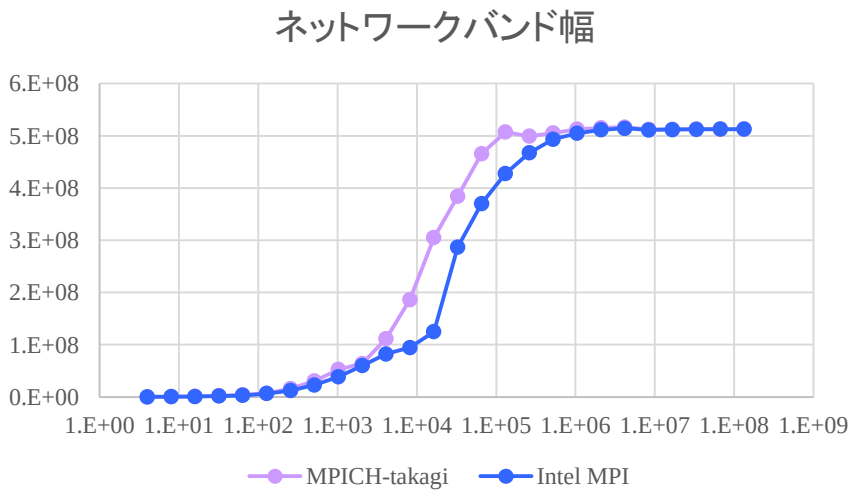
- 下記以外のLinux system callはLinuxに委譲
  *はシステムコールエントリとして存在するがダミー関数

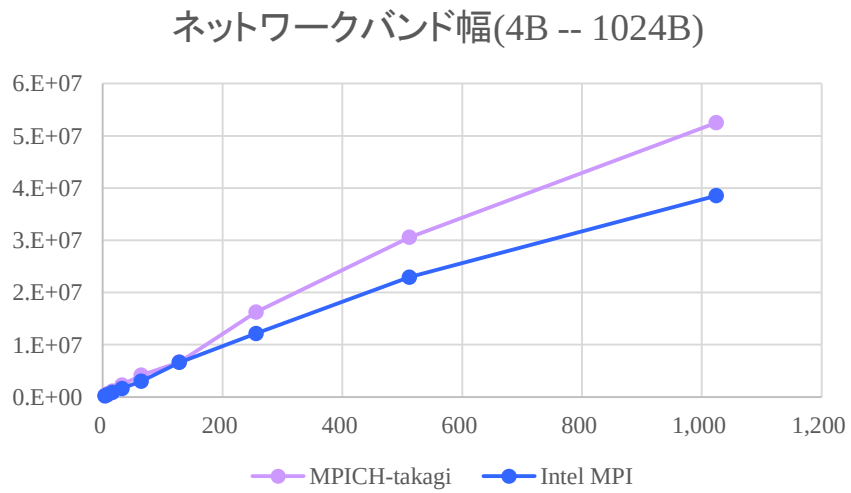
	実装済み	実装予定
Process Thread	clone, exit_group, exit, kill, tgkill, rt_sigaction, rt_sigprocmask, set_tid_address, getrlimit, getpid, arch_prctl, futex, set_robust_list	getuid, getgid, setuid, setgid, geteuid, getegid, setpgid, getpgrp, setsid, setreuid, setregid, getgroups, setgroups, setresuid, getresuid, setresgid, getresgid getpgid, setfsuid, setfsgid, getsid, setrlimit, gettid, set_thread_area, get_thread_area
Memory management	mmap, munmap, mprotect, brk, madvise*	mremap, remap_file_pages, mincore, shmget, shmat, shmctl, shmdt, mlock, munlock, mlockall, munlockall, modify_ldt, swapon*, swapoff*
Scheduling	sched_setaffinity*, sched_getaffinity*, sched_yield	nanosleep, getitimer, alarm, setitimer, gettimeofday, times, settimeofday, time, sched_setaffinity, sched_getaffinity
Performance Counter	独自インターフェイスpmc_init, pmc_start, pmc_stop, pmc_reset	PAPI対応

- ✓ McKernel上へのプロセス生成はLinuxからおこなう。現在McKernel上は1プロセスNスレッド。今後複数プロセス生成
- ✓ Mmapすべてのフラグを実装していない
- ✓ swap機能はない
- ✓ /sys, /procは、Linux側でXeon Phi環境をエミュレーション

- さらにMcKernel側で実装すべき機能は検討中

MPICHの移植状況


ネットワークバンド幅
6.E+08
5.E+08
4.E+08
3.E+08
2.E+08
1.E+08
0.E+00
1.E+00 1.E+01 1.E+02 1.E+03 1.E+04 1.E+05 1.E+06 1.E+07 1.E+08 1.E+09
MPICH-takagi
Intel MPI


ネットワークバンド幅(4B -- 1024B)
6.E+07
5.E+07
4.E+07
3.E+07
2.E+07
1.E+07
0.E+00
0
200
400
600
800
1,000
1,200
MPICH-takagi
Intel MPI

現状＆今後

- 2013年11月 : 1st リリース
  - IHK, McKernel, DCFA, MPICH/DCFA for Xeon Phi
  - IHK and McKernel for Xeon and vmware
- メーリングリスト
  - mckernel-users@pccluster.org立ち上げ
- Linux API機能検証実施中
  - LTP (Linux Test Project)利用
- IHK, McKernel内部ドキュメント作成中
- 今後
  - Linux API性能検証実施
  - McKernel OS Services実装
  - 2014年3月 : 2nd リリース
  - Xeon & Xeon PHI以外のアーキテクチャへの適用

関連論文発表

1. Masamichi Takagi, Yuichi Nakamura, Atsushi Hori, Balazs Gerofi, and Yutaka Ishikawa, "Revisiting Rendezvous Protocols in the Context of RDMA-capable Host Channel Adapters and Many-Core Processors," EuroMPI2013, 2013.
2. Balazs Gerofi, Akio Shimada, Atsushi Hori, Yutaka Ishikawa, “Partially Separated Page Tables for Efficient Operating System Assisted Hierarchical Memory Management on Heterogeneous Architectures,” CCGRID 2013, pp. 360-368, 2013.
3. Min Si, Yutaka Ishikawa, and Masamichi Takagi, "Direct MPI Library for Intel Xeon Phi co-processors," The 3rd Workshop on Communication Architecture for Scalable Systems (CASS 2013) in conjunction with IPDPS2013, 2013.
4. Min Si, Yutaka Ishikawa, “Design of Communication Facility on Heterogeneous Cluster ,” 情報処理学会、2012-HPC-133、2012.
5. Min Si, “An MPI Library Implementing Direct Communication for Many-Core Based Accelerators,” SC’12 poster, 2012
6. Yuki Matsuo, Taku Shimosawa, and Yutaka Ishikawa, "A File I/O System for Many-Core Based Clusters,” in conjunction with ICS2012, 2012.
7. Min Si and Yutaka Ishikawa, "Design of Direct Communication Facility for Manycore-based Accelerators, “ to appear at CASS2012 in conjunction with IPDPS2012, 2012
8. 下沢、石川、堀敦史、並木美太郎、辻田祐一、「メニーコア向けシステムソフトウェア開発のための実行環境の設計と実装」、情報処理学会、2011-OS-118(1), 1-7, 2011.
9. Taku Shimosawa, Yutaka Ishikawa, "Inter-kernel Communication between Multiple Kernels on Multicore Machines", IPSJ Transactions on Advanced Computing Systems Vol.2 No.4 (ACS 28), pp. 64-82, 2009.
10. Taku Shimosawa, Hiroya Matsuba, Yutaka Ishikawa, "Logical Partitioning without Architectural Supports", 32nd IEEE Intl. Computer Software and Applications Conference (COMPSAC 2008), pp. 355-364, 2008.