Towards Exascale Computing

Yutaka Ishikawa

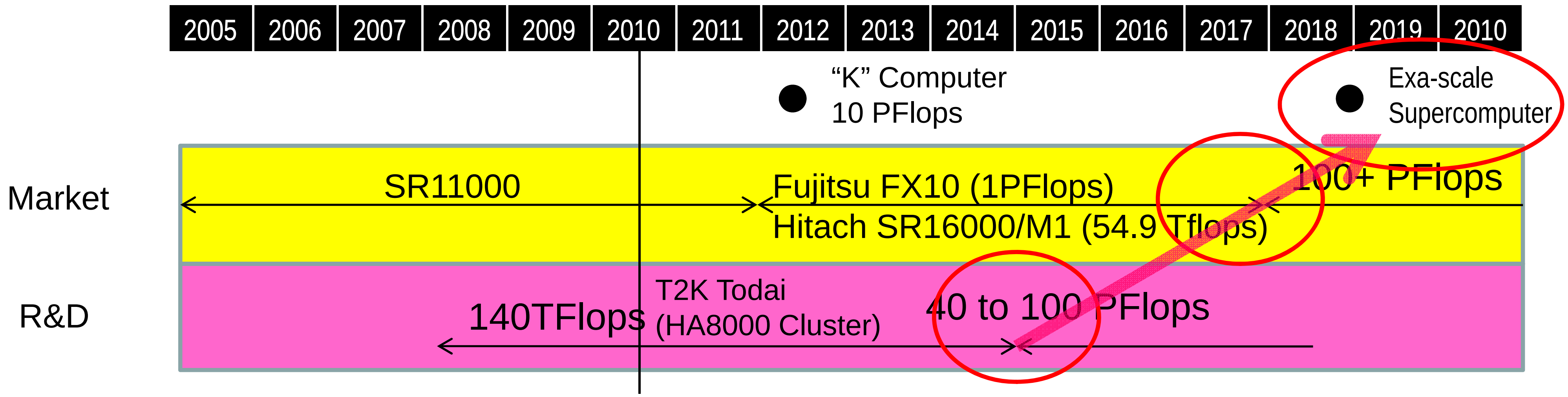
University of Tokyo

RIKEN AICS

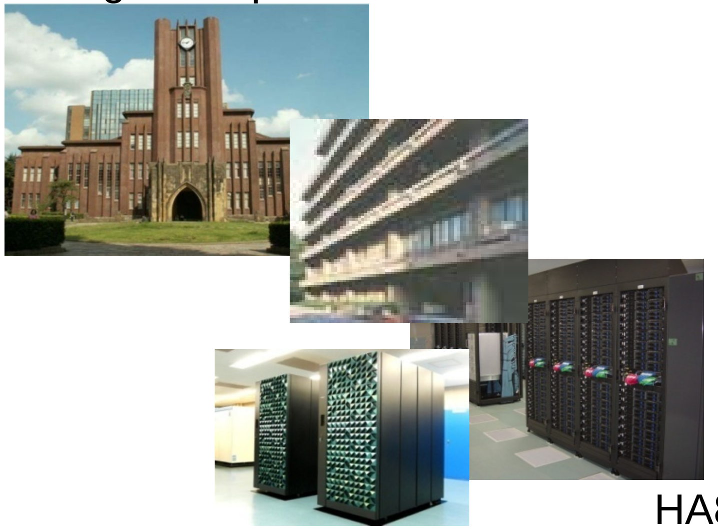
Outline of This Talk

- Activities in U. of Tokyo and Riken AICS
  - Many-core based PC Cluster
  - System Software Stack
  - Prototype System
- Rethinking of How to use MPI Library in state-of-the-art supercomputers
  - Are MPI_Isend/MPI_Recv really help for overlapping programming ?

Post T2K Todai



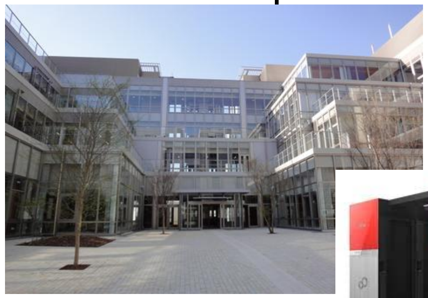
Hongo Campus



HA8000

SR16000/M1

Kashiwa Campus

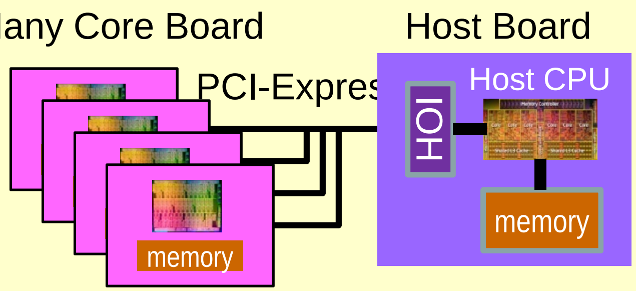


FX10

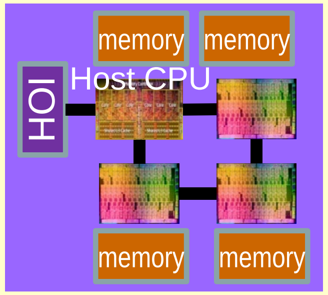
- PRIMEHPC FX10
  - 4800 Node (16 core/node)
  - 1.13 PFlops
  - 150 Tbyte Memory
- Hitachi SR16000/M1
  - 56 Node (32 core/node)
  - 54.9 TFlops
  - 11200 Gbyte Memory

Variations of Many-core based machines

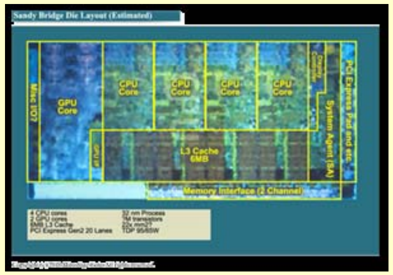
Many-core board connected to PCI-Express
e.g., Intel Knights Ferry, Knights Corner



Many-core chip connected to system bus
Not existing so far

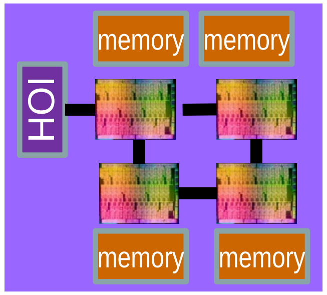


Many-core inside CPU die
c.f., Intel Sandy Bridge with GPU



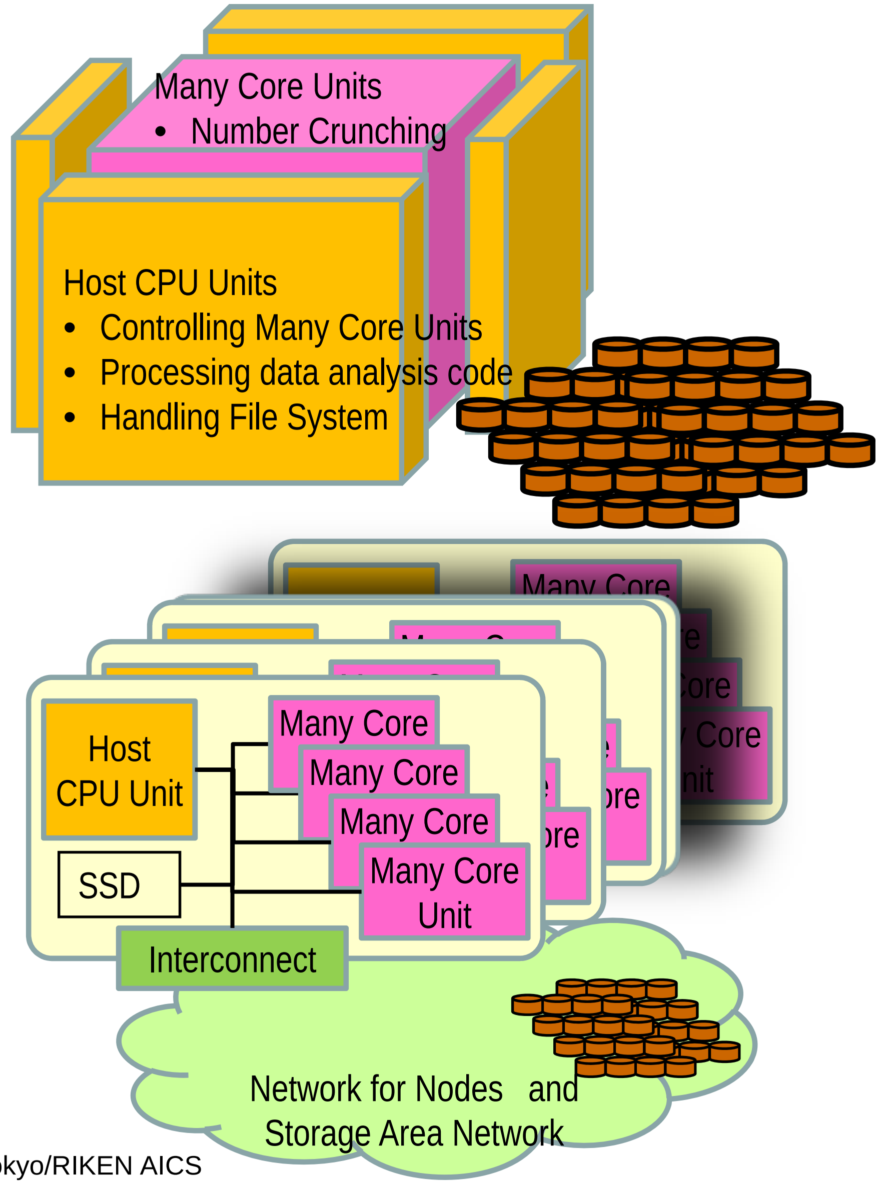
http://pc.watch.impress.co.jp/docs/column/kaigai/20100412_360173.html

Many-core only
Not existing fo far



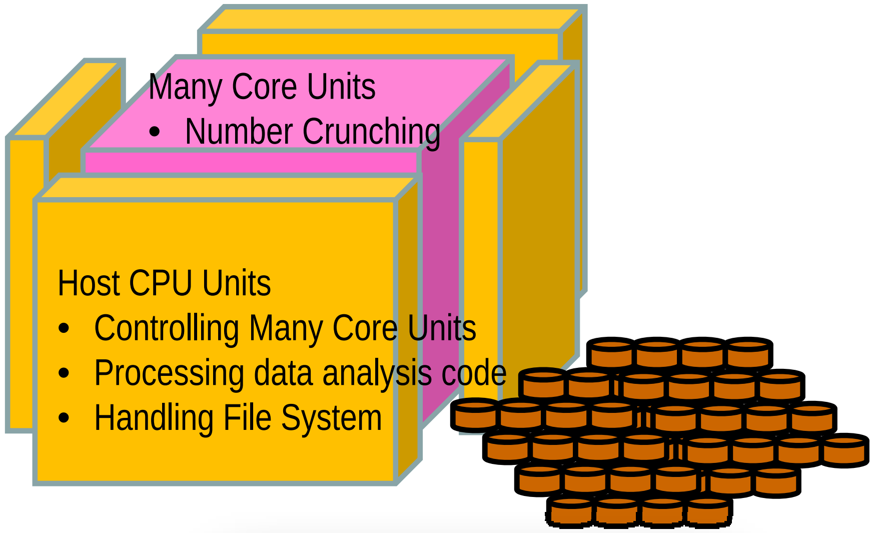
Post T2K System Image: Requirements

- Both requirements of large data analysis and number crunching applications must be satisfied.
  - Performance of I/O
  - Performance of floating point operations
  - Parallel Performance



Post T2K System Image: Execution Image

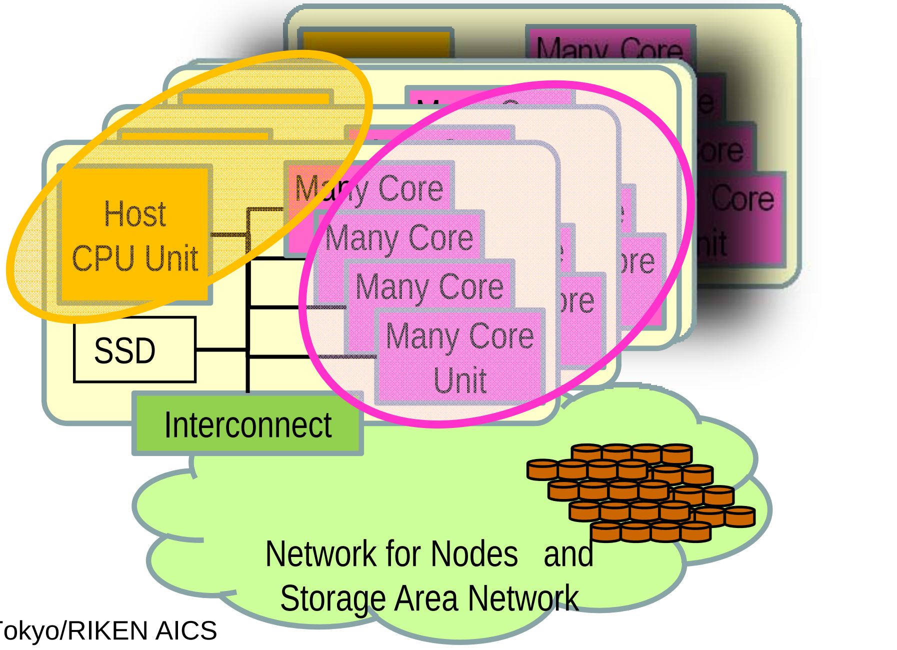
- Both requirements of large data analysis and number crunching applications must be satisfied.
  - Performance of I/O
  - Performance of floating point operations
  - Parallel Performance



Co-execution of 2 types of job within partition

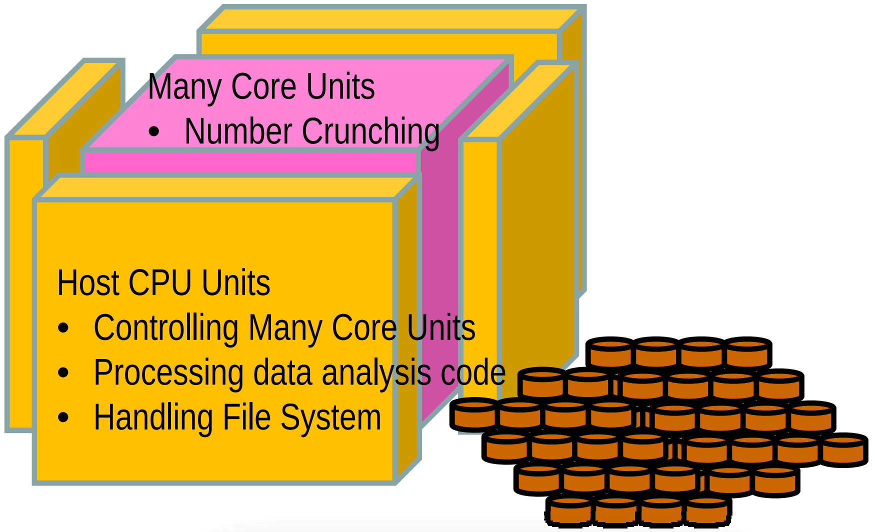
ManyCores: Number crunching application
Host CPU is used for file I/O and memory swap

Host CPUs: I/O intensive application



Post T2K System Image: Execution Image

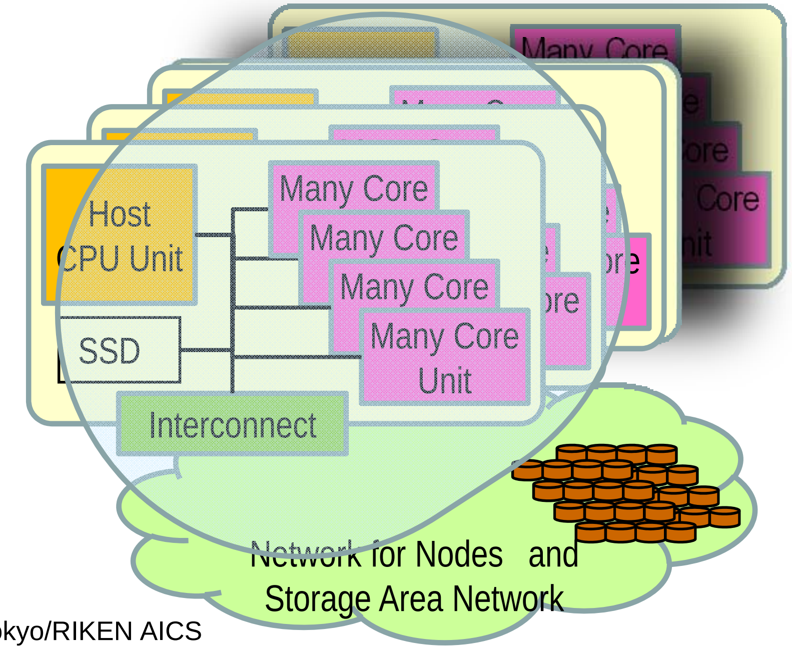
- Both requirements of large data analysis and number crunching applications must be satisfied.
  - Performance of I/O
  - Performance of floating point operations
  - Parallel Performance



One Job execution within partition

ManyCores: Computation and Communication

Host CPUs: Memory Share/Swap & Communication & I/O



Post T2K System Image: Execution Image

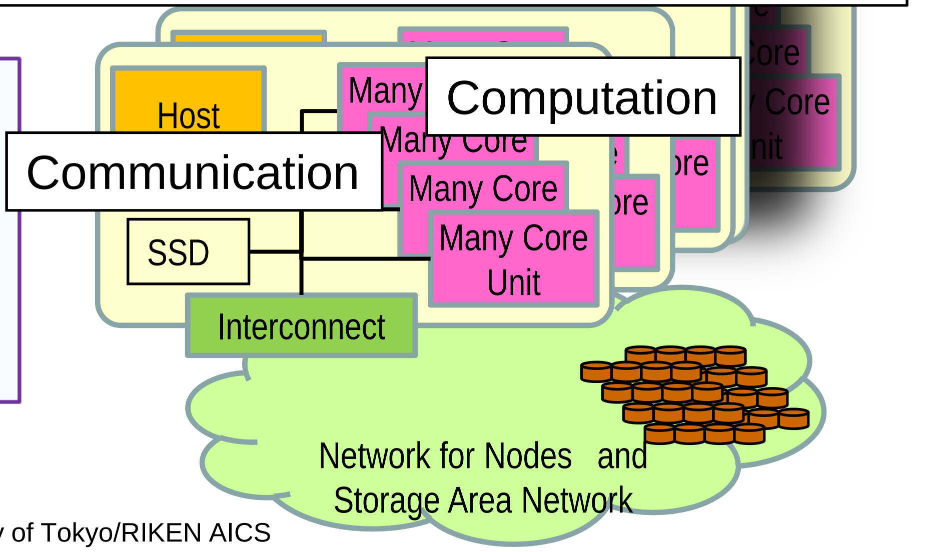
- Both requirements of large data and number crunching applicatio be satisfied.
  - Performance of I/O
  - Performance of floating poi operations
  - Parallel Performance

```
do {
      for (……) {
            for (…..) {
                  for (……) {
                              /* Computation */
                              /* Due to the limited memory in many
                              core units, data is swapped to memory
                              in Host CPU */
                  }
            }
      }
      /* Now many data is located in Host memory */
      /* Data exchange among remote node */
} while (…);
```

One Job execution within partition

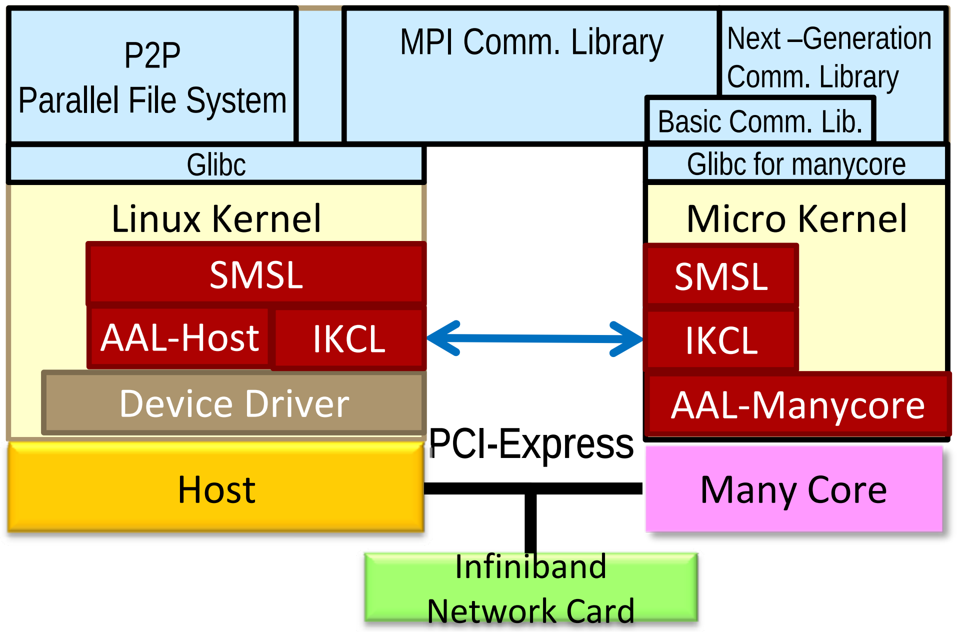
ManyCores: Computation and Communication

Host CPUs: Memory Share/Swap
& Communication & I/O

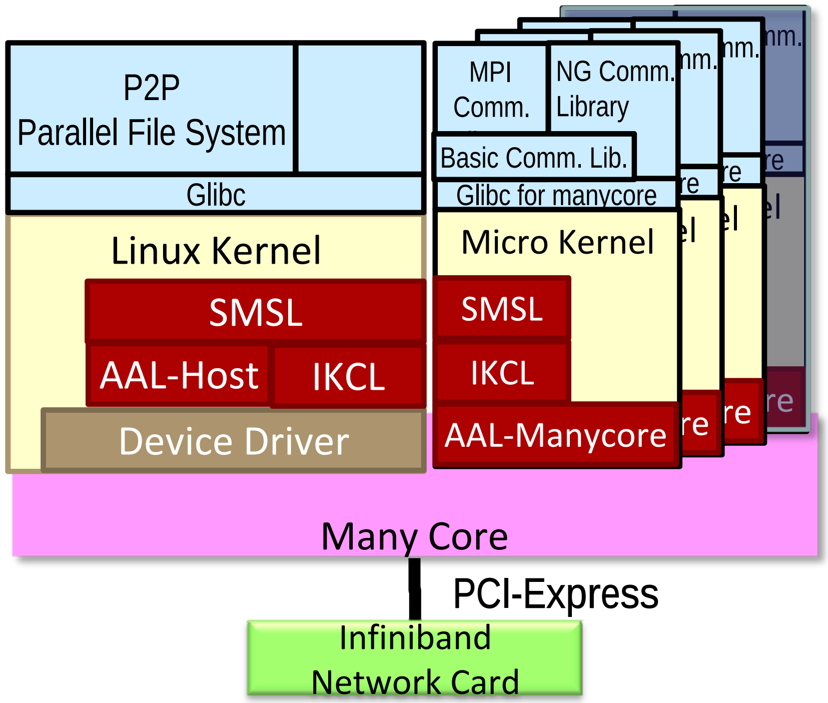


Post T2K System Software Stack

In case of Non-Bootable Many Core



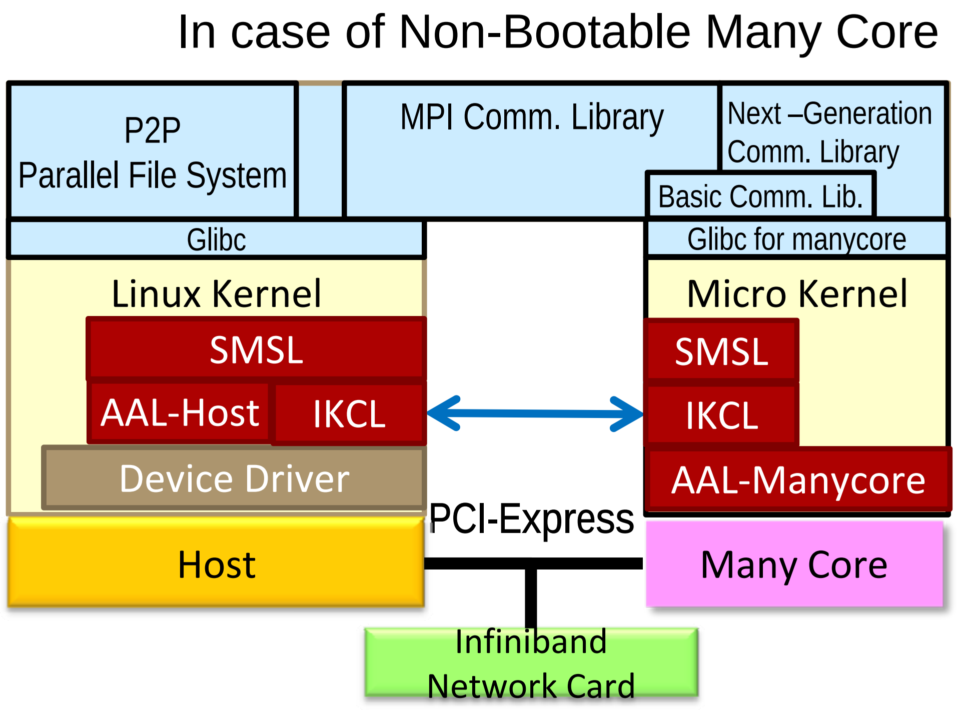
In case of Bootable Many Core

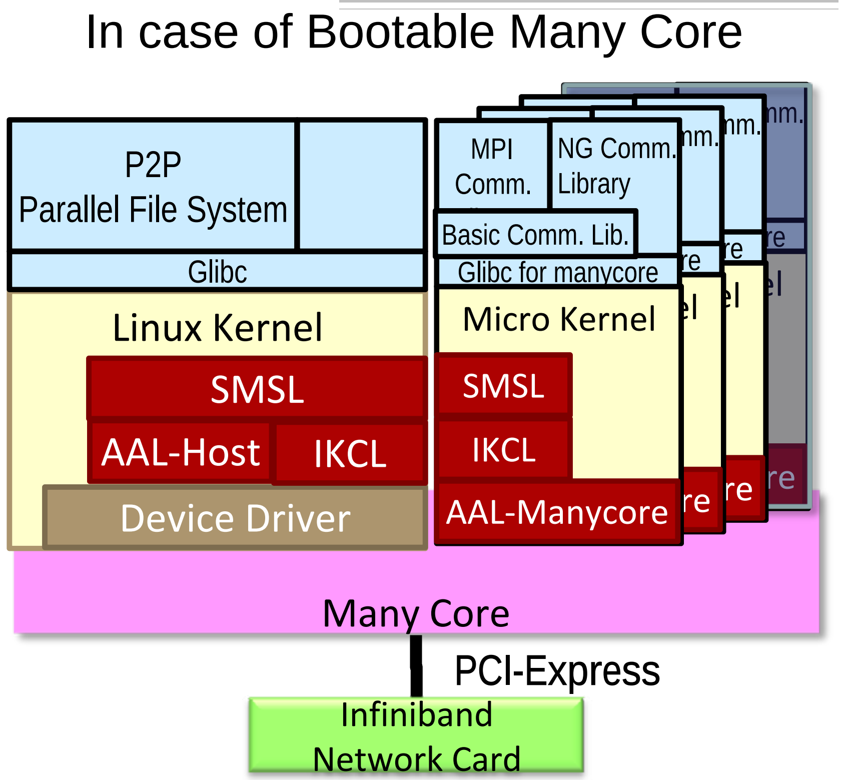


- AAL (Accelerator Abstraction Layer)
  - Provides low-level accelerator interface
  - Enhances portability of the micro kernel
- IKCL (Inter-Kernel Communication Layer)
  - Provides generic-purpose communication and data transfer mechanisms
- SMSL (System Service Layer)
  - Provides basic system services on top of the communication layer

Design Criteria
- Cache-aware system software stack
- Scalability
- Minimum overhead of communication facility
- Portability

Post T2K System Software Stack





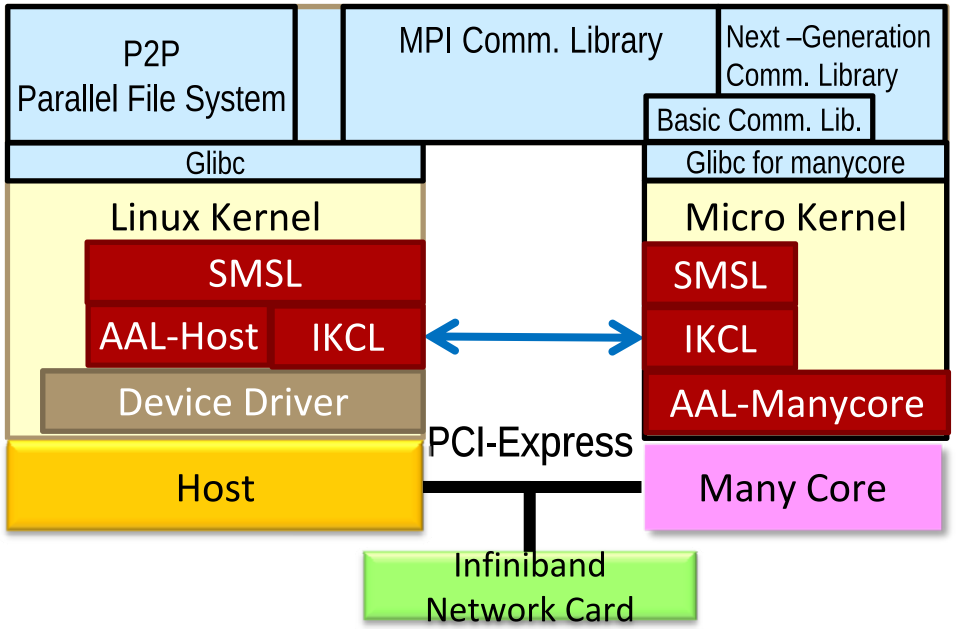
- AAL (Accelerator Abstraction Layer)
  –
  –
- IKCL
  – ata
- SMSL (System Service Layer)
  – Provides basic system services on top of the communication layer

Because manycores have small memory caches and limited memory bandwidth, the footprint in the cache during both user and system program executions should be minimized.

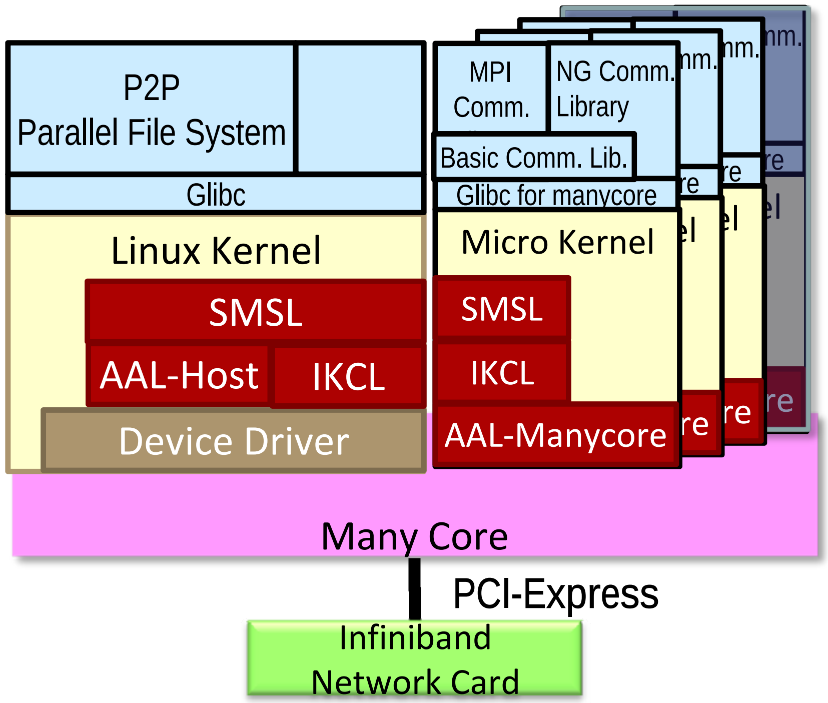
Design Criteria
- Cache-aware system software stack
- Scalability
- Minimum overhead of communication facility
- Portability

Post T2K System Software Stack

In case of Non-Bootable Many Core



In case of Bootable Many Core



- AAL (Accelerator Abstraction Layer)
  - Provides low-level accelerator interface

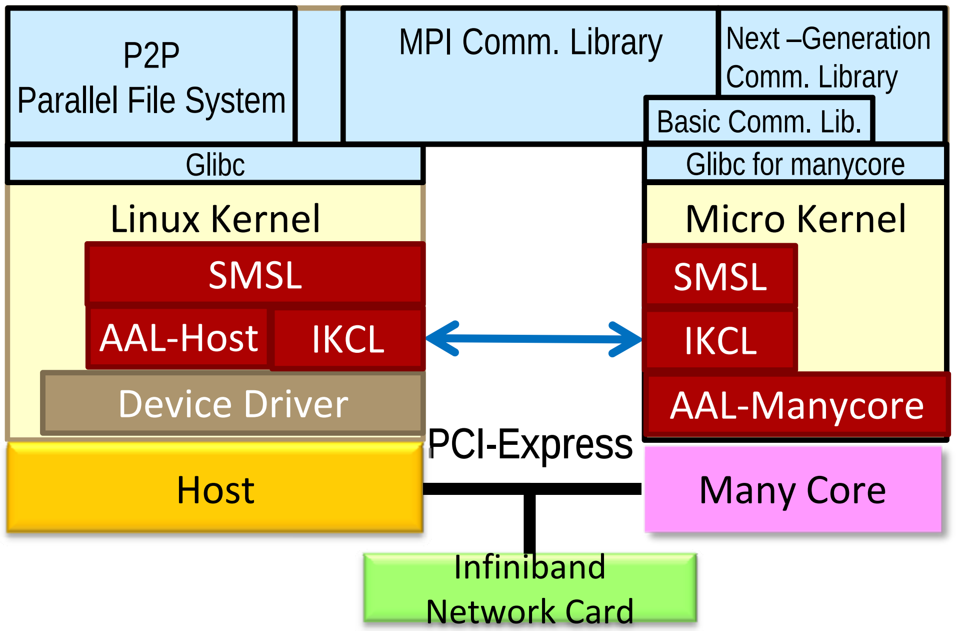
One of the scalability issues results from enlarging the internal data structures to manage resources for not only local node but also other nodes. A new resource management technique should be designed.

- I
- data
- S
  - Provides basic system services on top of the communication layer

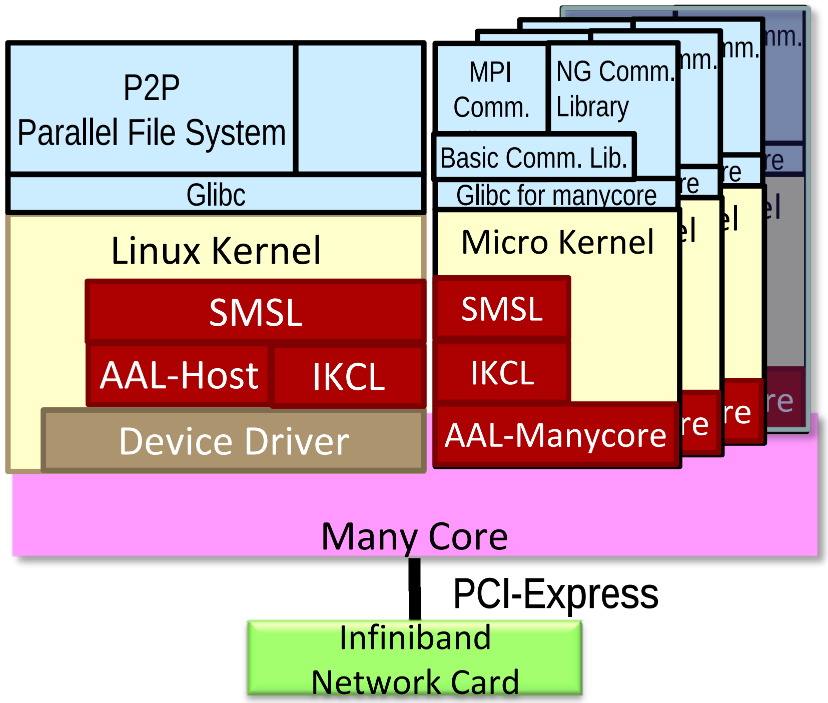
Design Criteria
- Cache-aware system software stack
- Scalability
- Minimum overhead of communication facility
- Portability

Post T2K System Software Stack

In case of Non-Bootable Many Core



In case of Bootable Many Core



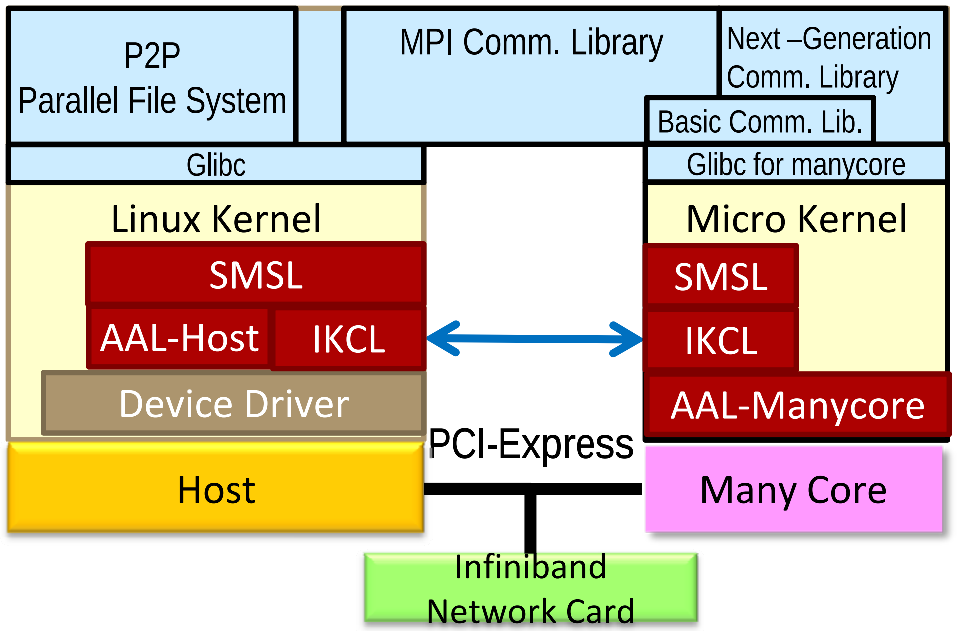
- AAL (Accelerator Abstraction Layer)
  - Provides low-level accelerator interface
  - Enhances portability of the micro kernel
- IK
  - ata
- S
  - Provides basic system services on top of the communication layer

Minimum overhead of communication between cores as well as direct memory access between manycore units is required for strong scaling.

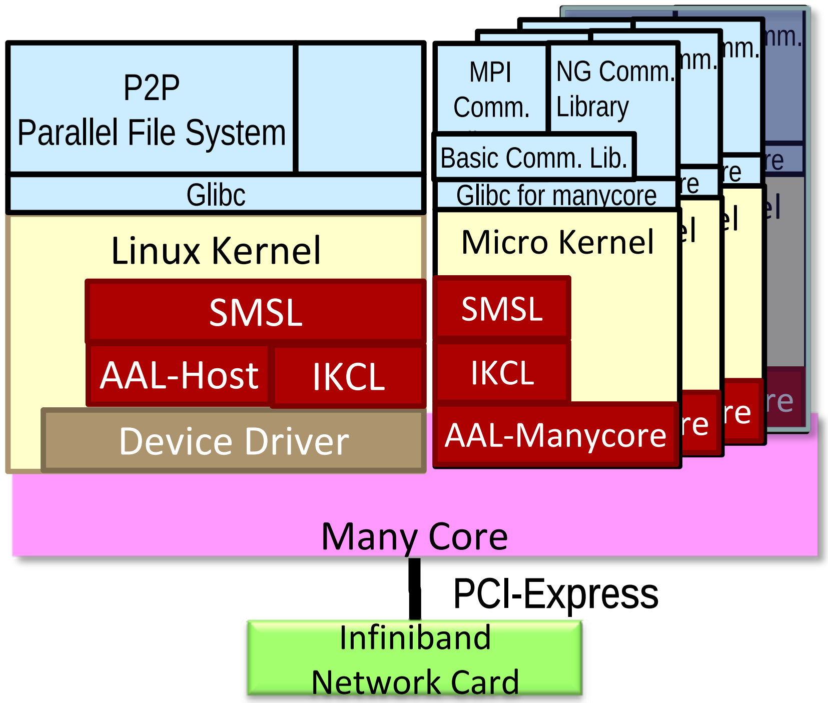
Design Criteria
- Cache-aware system software stack
- Scalability
- Minimum overhead of communication facility
- Portability

Post T2K System Software Stack

In case of Non-Bootable Many Core



In case of Bootable Many Core



- AAL (Accelerator Abstraction Layer)
  - Provides low-level accelerator interface
  - Enhances portability of the micro kernel
- IKCL (Inter-Kernel Communication Layer)
  - Provides generic-purpose communication and data
- SMSL
  - Provides basic system services on top of the communication layer

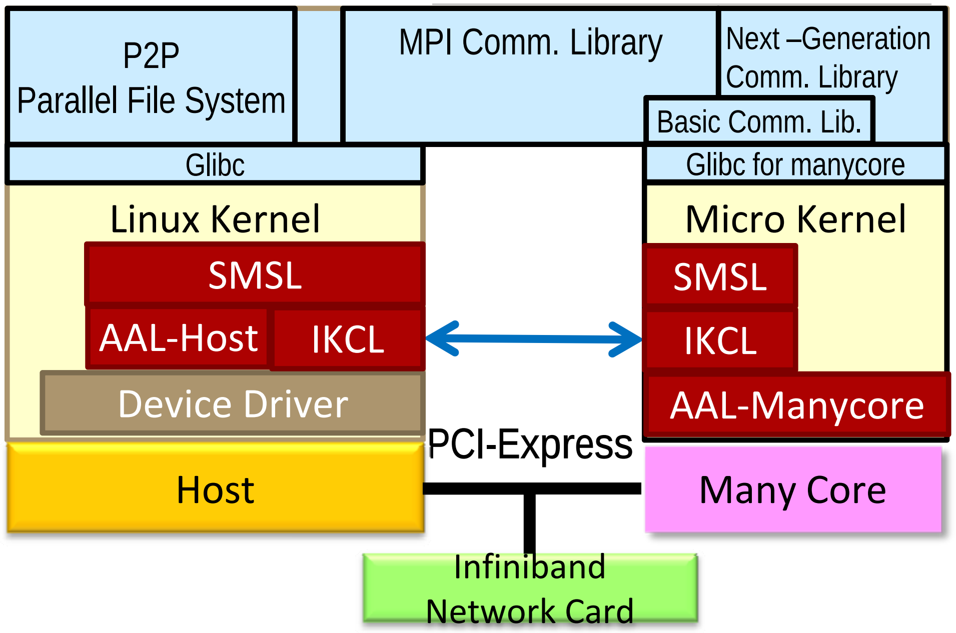
Easy software migration from cluster systems must be hold.

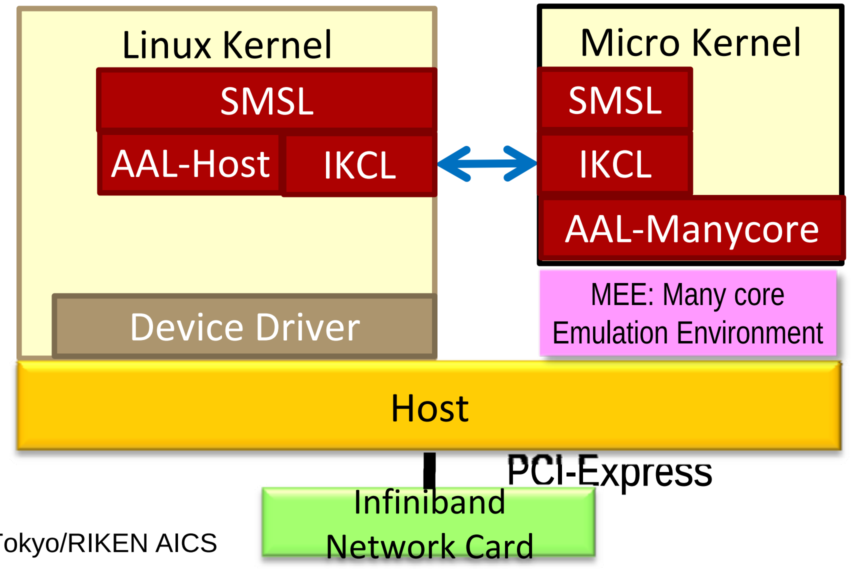
Design Criteria
- Cache-aware system software stack
- Scalability
- Minimum overhead of communication facility
- Portability

Current Status

- SMSL, AAL, and IKCL
  - Taku Shimosawa (Ph.D student)
- HIDOS Prototype Kernel
  - Taku Shimosawa (Ph.D student)
- Direct Communication in MIC
  - Min Si (Master student)
- Paging system and file I/O
  - Yuki Matsuo (Bachelor student)

- MEE: Many Core Emulation Environment for developers who cannot access MIC
  Taku Shimosawa (Ph.D student)

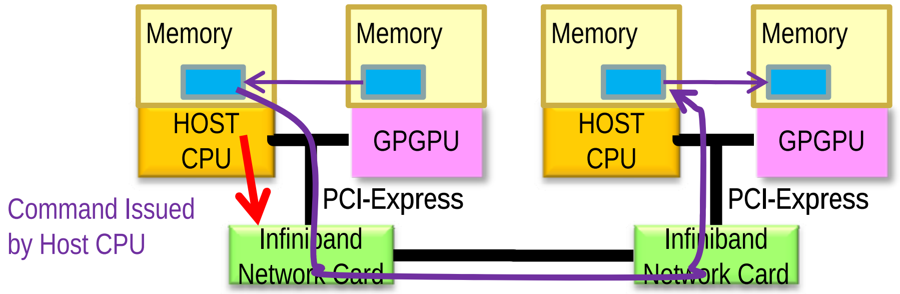




DCFA: Direct Communication Facility for Accelerator

Communication in GPU

- An accelerator is a PCI-Express device, and thus it cannot configure/initialize another device such as a communication device
- Though the PCI-Express address is known by a GPU, the GPU cannot issue commands to a communication device.
- The Mellanox GPU Direct technology does not provide direct communication between GPUs, but data is copied to memory in Host CPU and then transferred to remote Host, …



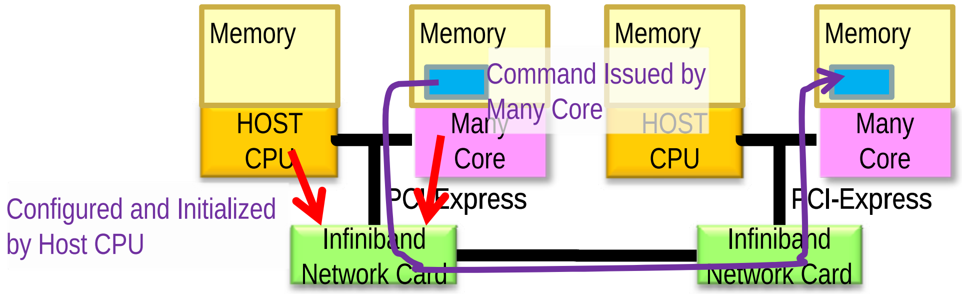
Communication in MIC

- If MIC knows the PCI-Express address of a communication device, it may issue commands to that device.
- However, MIC cannot receive signals from PCI-Express devices.

DCFA

(Direct Communication Facility for Accelerator)

Designed and implemented at U. of Tokyo



Rethinking of MPI Library Usage in state-of-the-art Supercomputers

- Misunderstanding the semantics of MPI_Isend / MPI_Irecv primitives

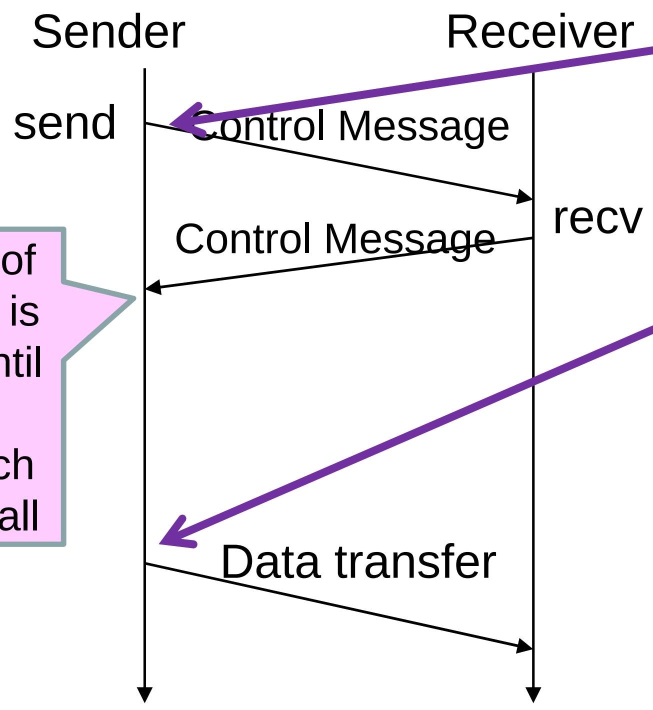
```
for (…..) {
    /* Computation */
    MPI_Irecv(buf, count, MPI_DOUBLE, src, tag, MPI_
                  COMM_WORKD, &req[0]);
    MPI_Isend((buf, count, MPI_DOUBLE, dest, tag, M
                  PI_COMM_WORLD, &req[1]);

    /* Computation */

    MPI_Waitall(2, req, stat);
}
```

The programmer thinks that communication and computation are overlapping

Rethinking of MPI Library Usage in state-of-the-art Supercomputers

- Misunderstanding the semantics of MPI_Isend / MPI_Irecv primitives

Rendezvous Protocol



Progression of data transfer is postponed until calling MPI functions such as MPI_Waitall

```
for (…..) {
    /* Computation */
    MPI_Irecv(buf, count, MPI_DOUBLE, src, tag, MPI_
              COMM_WORKD, &req[0]);
    MPI_Isend((buf, count, MPI_DOUBLE, dest, tag, M
              PI_COMM_WORLD, &req[1]);
    /* Computation */
    MPI_Waitall(2, req, stat);
}
```

Eager Protocol

- When a message send primitive is posted, the message is immediately sent to the receiver

    Pros: small latency

    Cons: If the message size is large and the receiver has not posted a receive, a large buffer has to be created and copy data.

Rendezvous Protocol

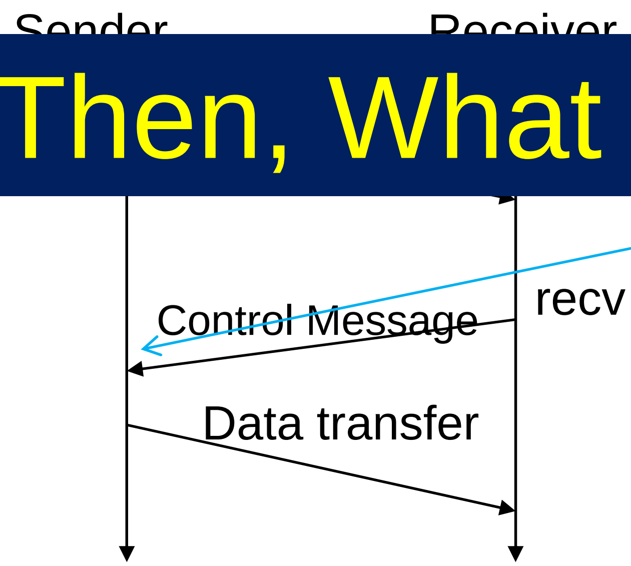
- See the left-hand side figure

Rethinking of MPI Library Usage in state-of-the-art Supercomputers

- Misunderstanding the semantics of MPI_Isend / MPI_Irecv primitives

```
for (…..) {
     /* Computation */
     MPI_Irecv(buf, count, MPI_DOUBLE, src, tag, MPI_
                    COMM_WORKD, &req[0]);
     MPI_Isend((buf, count, MPI_DOUBLE, dest, tag, M
                                        , &req[1]);

     MPI_Waitall(2, req, stat);
}
```

Rendezvous Protocol



Then, What we should do ?

Progression of data transfer is postponed until calling MPI functions such as MPI_Waitall

Eager Protocol

- When a message send primitive is posted, the message is immediately sent to the receiver

  Pros: small latency

  Cons: If the message size is large and the receiver has not posted a receive, a large buffer has to be created and copy data.

Rendezvous Protocol

- See the left-hand side figure

Rethinking of MPI Library Usage in state-of-the-art Supercomputers

- Persistent Communication

```
MPI_Recv_init(buf, count, MPI_DOUBLE, src, tag,
                        MPI_COMM_WORLD, &req[1]);
MPI_Send_init(buf, count, MPI_DOUBLE, dest, tag,
                        MPI_COMM_WORKD, &req[0]);
for  (I = 0; …….) {
   /` Computation `/
   MPI_Startall(2, req);

   /* Computation */

   MPI_Waitall(2, req, stat);
}
```

```
for (…..) {
     /* Computation */
     MPI_Irecv(buf, count, MPI_DOUBLE, src, tag, MPI_
                     COMM_WORKD, &req[0]);
     MPI_Isend((buf, count, MPI_DOUBLE, dest, tag, M
                     PI_COMM_WORLD, &req[1]);

     /* Computation */

     MPI_Waitall(2, req, stat);
}
```

MPI_Recv_init, MPI_Send_init:
Initialization of send and receive points. The request structures returned by those functions are used to issue actual communication

MPI_Start/MPI_Startall:
Issue actual communication specified by request structures

Rethinking of MPI Library Usage in state-of-the-art Supercomputers

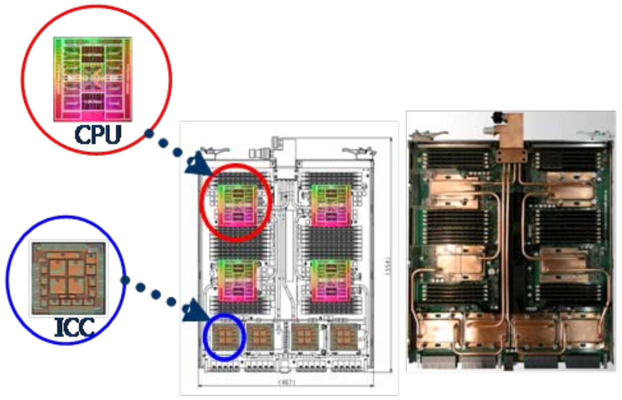
- Persistent Communication

```
MPI_Recv_init(buf, count, MPI_DOUBLE, src, tag,
                    MPI_COMM_WORLD, &req[1]);
MPI_Send_init(buf, count, MPI_DOUBLE, dest, tag,
                    MPI_COMM_WORKD, &req[0]);
for  (I = 0; …….) {
   /` Computation `/
   MPI_Startall(2, req);

   /* Computation */

   MPI_Waitall(2, req, stat);
}
```

MPI_Recv_init, MPI_Send_init:
    Initialization of send and receive points. The request structures returned by those functions are used to issue actual communication
MPI_Start/MPI_Startall:
    Issue actual communication specified by request structures

Since all communication patterns have been known at the MPI_Start/MPI_Startall, we can utilize communication hardware

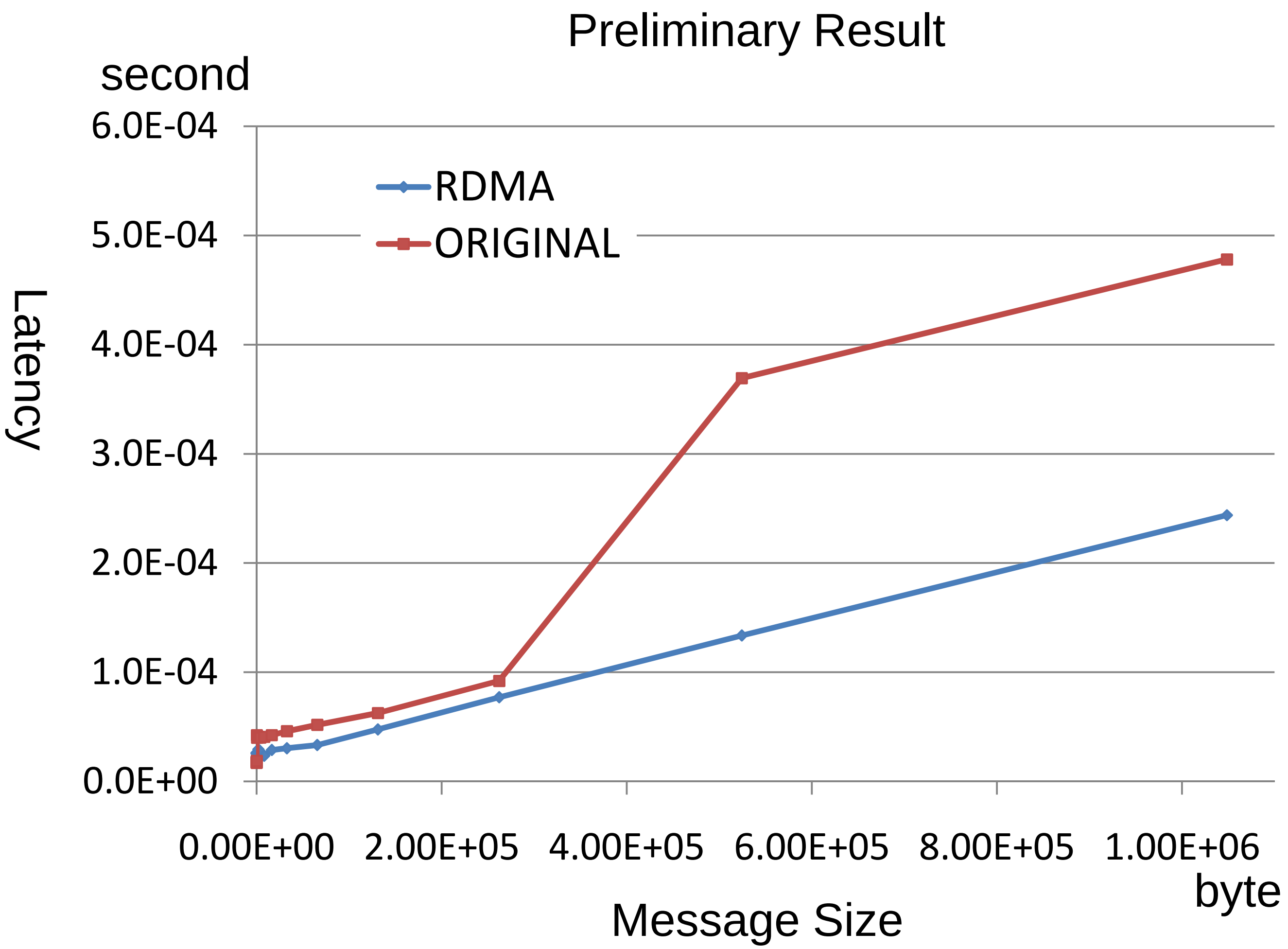
In K computer and Fujitsu FX-10, commercialize version of K, there are 4 DMA engines for communication



Low-latency RDMA based communication is realized in the persistent communication feature

Rethinking of MPI Library Usage in state-of-the-art Supercomputers

- Persistent Communication



MPI_Send_init, MPI_Recv_init, MPI_Start, MPI_Startall, MPI_Wait, MPI_Waitall are replaced using the MPI profiling feature.

The same program ran in both the RDMA-based implementation and the original one

Summary

- Activities in U. of Tokyo and Riken AICS
  - Many-core based PC Cluster
  - System Software Stack
  - Prototype System
- Rethinking of How to use MPI Library in state-of-the-art supercomputers
  - Are MPI_Isend/MPI_Recv really help for overlapping programming ?
  - Persistent Communication should be used instead of MPI_Isend/MPI_Recv

Strategic Direction/Development of HPC in JAPAN

Report on
Strategic Direction/Development of HPC
in JAPAN

今後のHPCI技術開発に関する報告書

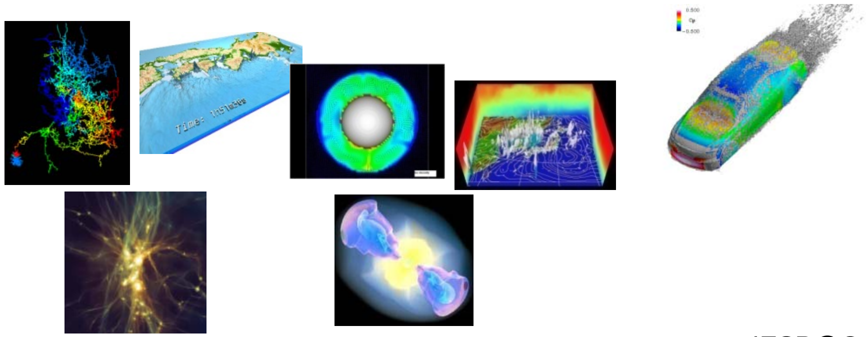
MEXT

MEXT: Ministry of Education, Culture, Sports, Science, and Technology

Council on HPCI Plan and Promotion

WG for Applications
Chairmen: Hirofumi Tomita (RIKEN AICS)
Junichiro Makino (TITECH)
Field 1: Life science/Drug manufacture
Field 2: New material/energy creation
Field 3: Global change prediction for disaster prevention/mitigation
Field 4: *Mono-zukuri* (Manufacturing technology)
Field 5: The origin of matters and the universe

WG for HPC Systems
Chairmen: Yutaka Ishikawa (U. of Tokyo/RIKEN AICS)
Subleaders:
Naoya Maruyama (TITECH), Masaaki Kondo (Elec. …) Yasuo Ishii (NEC), Akihiro Nomura (TITECH), Hiyoyuki Takizawa (Tohoku U.), Reiji Suda (U. of Tokyo), Takashiro Katagiri (U. of Tokyo)
Advisers:
Satoshi Matsuoka (TITECH), Kei Hiraki (U. of Tokyo), Hiroshi Nakamura (U. of Tokyo), Taisuke Boku (U. of Tsukuba), Atsushi Hori (RIKEN AICS), Mitaro Namiki (..), Shinji Sumimoto (Fujitsu), Hiroshi Nakashima (Kyoto U.) Mitsuhisa Sato (Tsukuba U.), Akinori Yonezawa (RIKEN AICS), Kengo Nakajima (U. of Tokyo), Ryutaro Himeno (RIKEN), Satoshi Sekiguchi (AIST), Kimihiko Hirao (RIKEN AICS), Akira Ukawa (U. of Tsukuba)



Strategic Direction/Development of HPC in JAPAN

Report on
Strategic Direction/Development of HPC
in JAPAN

今後のHPCI技術開発に関する報告書

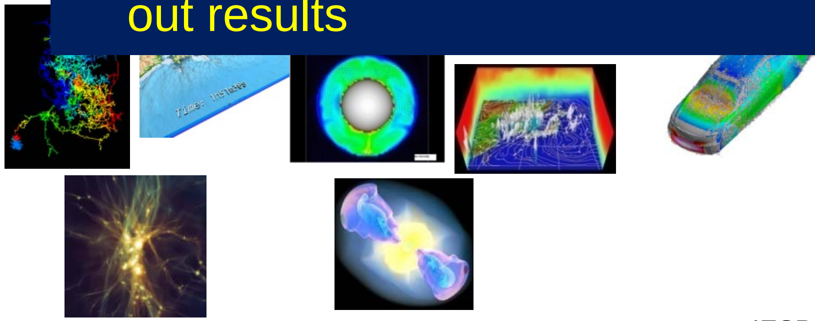
MEXT

Council on HPCI Plan ar
Promotion

During summer to winter in 2011:
Three times Joint meetings of WGs:
368 attendees in total

WG for Applications
Chairmen: Hirofumi Tomita (RIKEN AICS)
Junichiro Makino (TITECH)
Field 1: Life science/Drug manufacture

- Discussing science roadmap to challenge respective key socio-scientific problems in Japan by year 2020
- Demands for HPC systems to carry out results



WG for HPC Systems
Chairmen: Yutaka Ishikawa (U. of Tokyo/RIKEN AICS)
Subleaders:
Naoya Maruyama (TITECH), Masaaki Kondo (Elec. …)

- Studying key technologies to build an HPC system by 2018 to achieve the science roadmap, whose constraints are 2000 m^2 space and 20 to 30 MW electricity
  - Architectures, operating systems, middleware, programming model sand languages, math libraries

(RIKEN AICS), Akira Ukawa (U. of Tsukuba)

Required Systems to carry out Science Results by 2020

- Four types of process or architectures have been considered
  - General purpose (GP)
  - Capacity-bandwidth oriented (CB)
  - Reduced-memory (RM)
  - Throughput-oriented (TP)

e.g., Vector machines

e.g., using SOC

e.g., GPU

- Projection of 2018's systems if industries continue to develop their technologies without driving national projects

  Constraints:

    20 to 30MW electricity

    2000m^2 space

CPU

	Total CPU Performance PetaFLOPS	Total Memory Bandwidth PetaByte/s	Total Memory Capacity PetaByte
GP	200~400	20~40	20~40
CB	50~100	50~100	50~100
RM	500~1000	250~500	0.1~0.2
TP	1000~2000	5~10	5~10

Network

	Injection	P-to-P	Bisection	Minlatency	Max latency
High-radix (Dragonfly)	32 GB/s	32 GB/s	2.0 PB/s	200 ns	1000 ns
Low-radix (4D Torus)	128 GB/s	16 GB/s	0.13 PB/s	100 ns	5000 ns

Storage

Total Capacity	**Total Bandwidth**
1 EB	10TB/s
100 times larger than main memory	Bandwidth to save all data in main memory to disks within 1000 seconds

Source: "Report on Strategic Direction/Development of HPC"

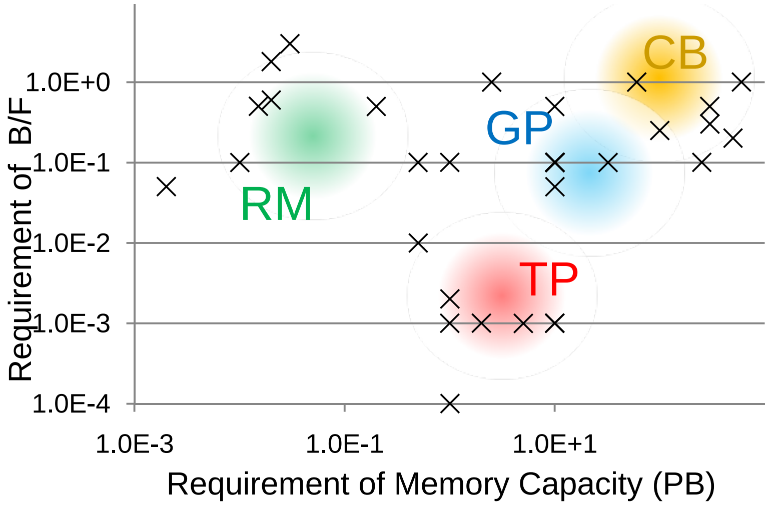
Required Systems to carry out Science Results by 2020

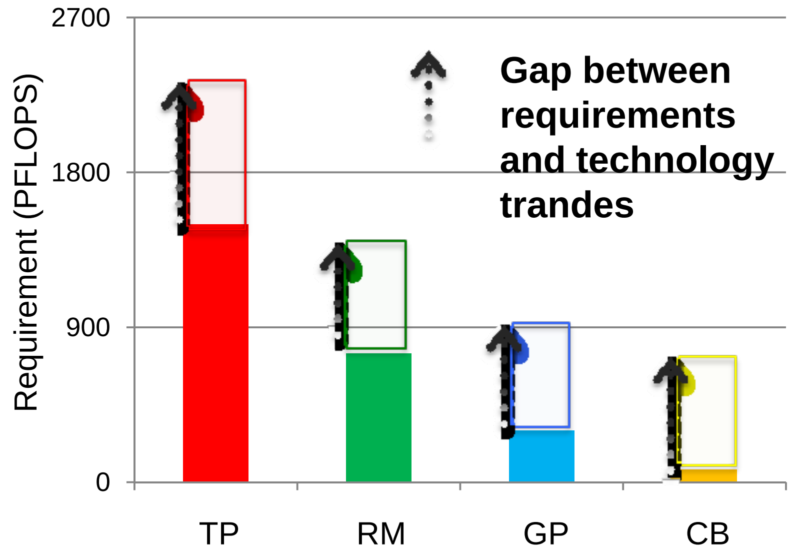
- Four types of processor architectures have been considered
  - General purpose (GP) — e.g., Vector machines
  - Capacity-bandwidth oriented (CB)
  - Reduced-memory (RM) — e.g., using SOC
  - Throughput-oriented (TP) — e.g., GPU
- Projection of 2018's systems if industries continue to develop their technologies without driving national projects

  Constraints:

  20 to 30MW electricity, 2000m^2 space
- Preliminary investigation on performance gap between 2018's systems and application requirements
  - A detailed report will be available in the end of March

Source: "Report on Strategic Direction/Development of HPC"





Concluding Remarks

- Two-year Feasibility Study will start at FY2012
  - About three groups will be selected
- After that, the government will decide developments