

超级计算机富岳 性能解决方案

富士通

Linux开发部 张雷

zhang.lei@fujitsu.com

2021.10.24

- 自我介绍
- 富岳系统简介
- 富岳软件系统简介
- 性能解决方案的介绍

自我介绍

- 中国 北京出生。 2005年来日留学
- 2008年进入富士通公司
- 2011年11月 参与PRIMEHPC FX10/FX100/富岳的开发工作。主要负责富士通芯片的驱动开发



富岳系统简介

为什么需要富岳超级计算机

- 为解决现代社会的各种各样的问题和科学技术方面的重要问题提供支持
- 为大数据和人工智能等需要海量计算的技术提供支持

e. g. 富岳为TOKYO2020奥运会的实时天气预报和新冠肺炎的研究都做出了积极的贡献






Design targets and approaches for Fugaku

■ 富岳名字的由来

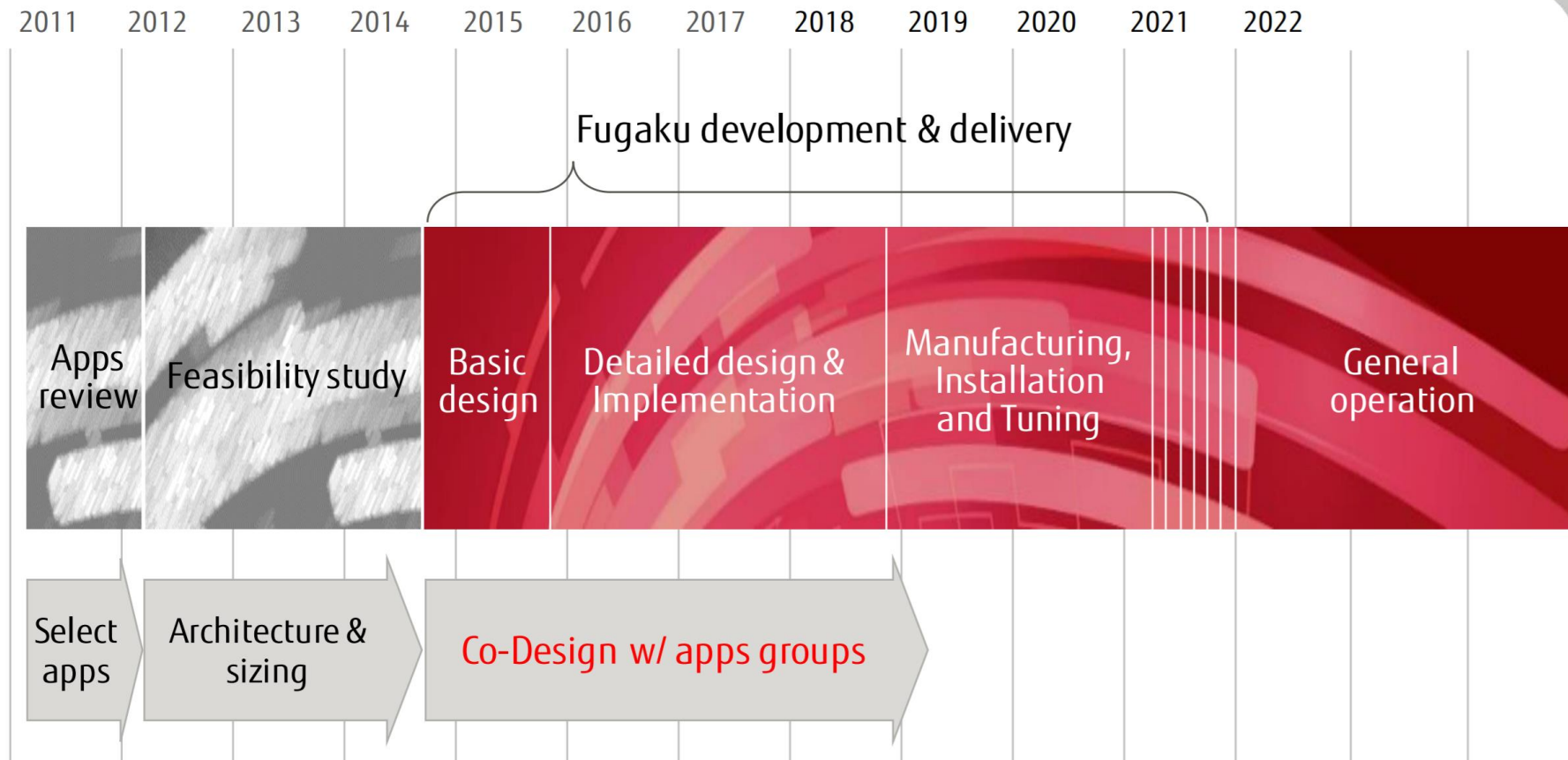
富岳是理化学研究所面向全世界募集的名字

富岳是富士山的别名。富士山是日本第一高山，并且山麓很广。所以这个命名包含高性能和广泛的应用两个意思。



Target	Approach
 Application performance	Co-design w/ application developers and Fujitsu-designed CPU core w/ high memory bandwidth utilizing HBM2
 Power efficiency	Leading-edge Si-technology, Fujitsu's proven low power & high performance logic design, and "Power Knobs"
 Usability	Armv8-A ISA with Scalable Vector Extension ("SVE"), and Arm standard Linux

富岳开发计划

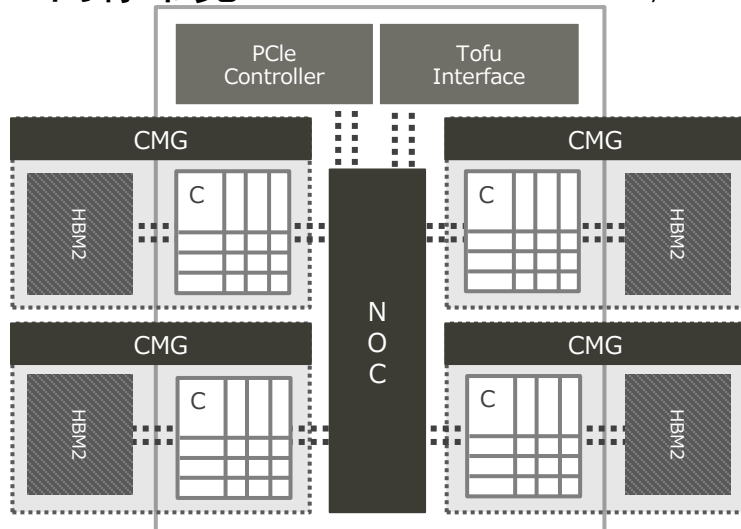


富岳采用的A64FX CPU的介绍

■ A64FX是Arm架构，首次采用SVE (Scalable Vector Extension) 技术的高性能，高效率的CPU

■ 双精度计算性能 >2.7 TFLOPS, >90%@DGEMM

■ 内存带宽 1024 GB/s, >80%@STREAM Triad



CMG : Core Memory Group NOC : Network on Chip

	A64FX
ISA (Base, extension)	Armv8.2-A, SVE
Process Technology	7 nm
双精度峰值性能	>2.7 TFLOPS
SIMD长度	512-bit
核数	48 + 4/2
内存容量	32 GiB (HBM2 x4)
内存带宽	1024 GB/s
PCIe	Gen3 16 lanes
网络连接	TofuD integrated

获奖经历

■ TOP500 (LINPACK)

- 442.01PFLOPS (2021年6月) 1位
- 约是第2位的Summit (148.6PFLOPS) 的3倍性能

■ HPCG

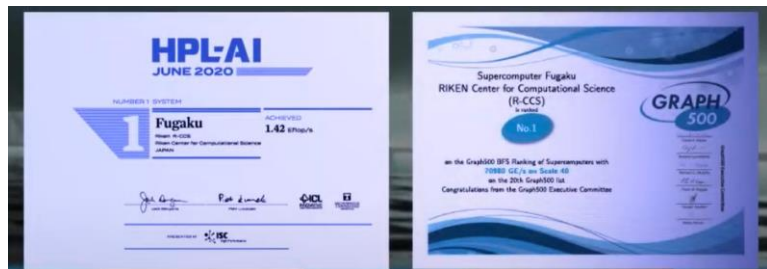
- 16.00PFLOPS (2021年6月) 1位
- 约是第2位のSummit (2.93PFLOPS) 的5.5倍性能

■ HPL-AI

- 2.004EFLOPS (2021年6月) 1位
- 约是第2位的Summit (1.15EFLOPS) 的1.7倍性能

■ Graph500

- 102,955GTEPS (traversed edges per second) (2021年6月) 1位
- 第2位的Sunway TaihuLight (23,756GTEPS) 的4倍性能



从2020年6月开始TOP500、HPCG、HPL-AI、Graph500连续三期获得世界第一

富岳系统软件介绍

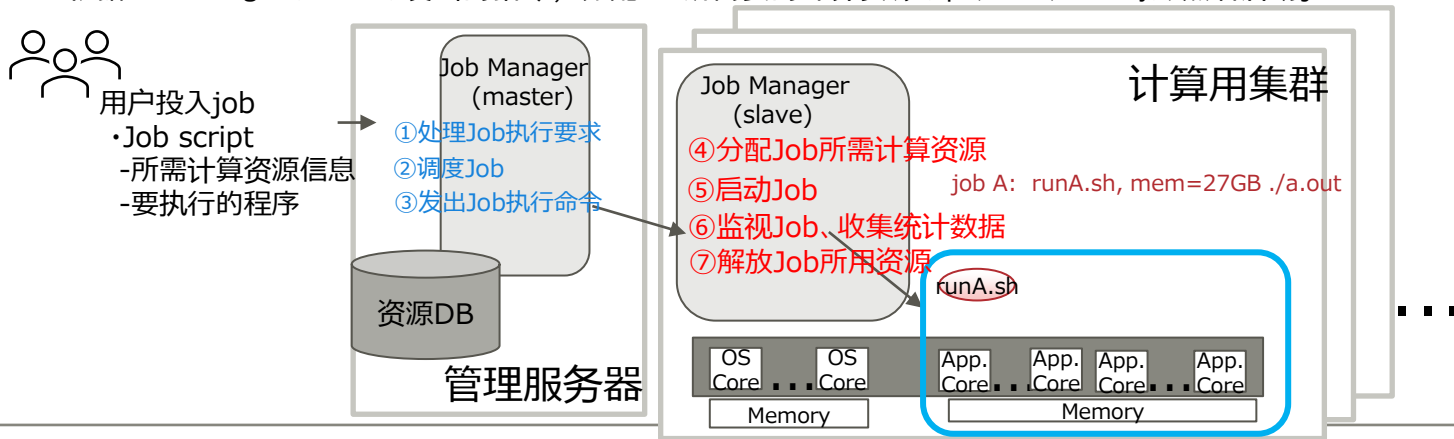
如何使用超级计算机

■ 如何使用超级计算机

- 为了让多个用户，共同的高效的使用超级计算机，我们采用批处理系统。批处理是大多数HPC大规模集群的程序执行方式。
- 批处理系统是各个用户把自己要执行的程序（job）投入到执行队列里面，然后依次执行。

■ JoB管理软件的构成

- 管理服务器: Job Manager (master)
 - Job执行要求的处理
 - Job调度管理。
 - 向计算用集群的Job Manager (slave) 发出执行Job的命令
- 计算用集群: Job Manager (slave)
 - 根据Job Manager (master) 发出的指令，分配Job所需要的计算资源(cpu, core, memory), 然后启动job



■ 对应用户多样的需求

- 采用RHEL8 (ARM), 使得各种各样的ISV/OSS应用程序可以执行
- 支持LWK (Light Weight Kernel)/docker的Job执行环境

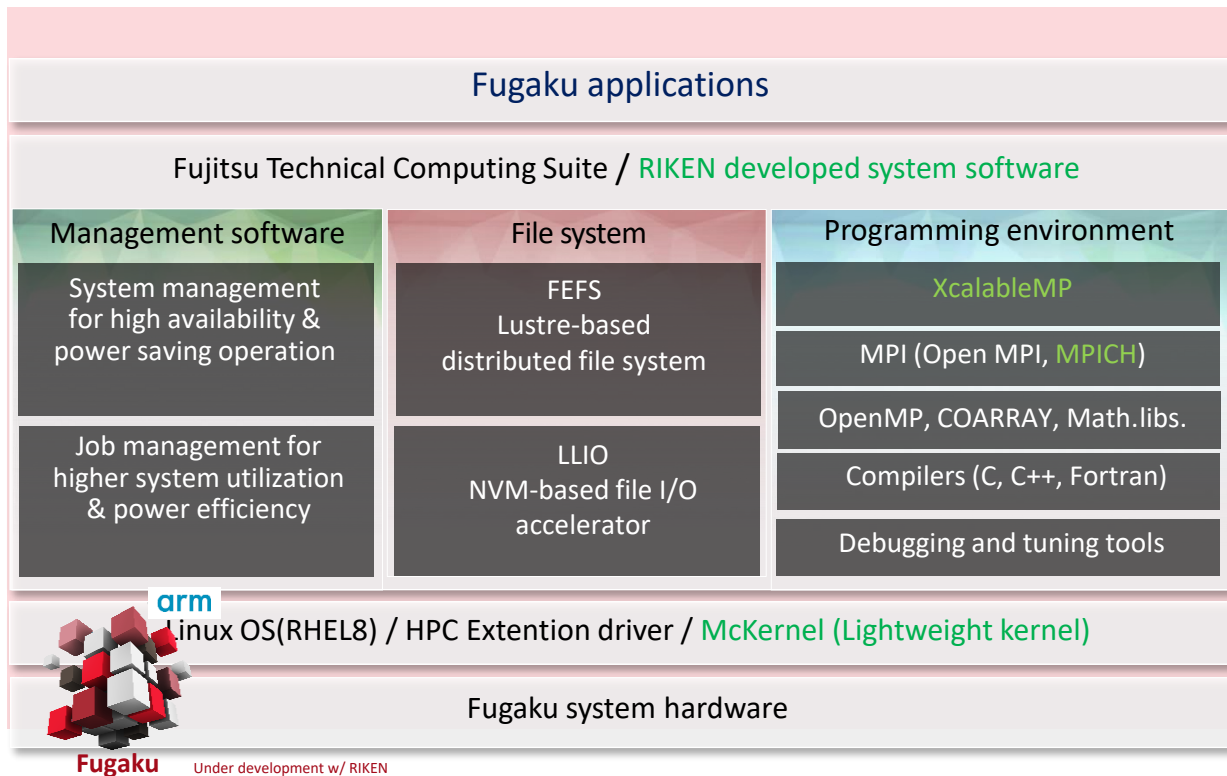
■ 支持大规模集群管理, Job管理的各种需求

- 可根据电力最大使用量进行Job调度
- 支持Job执行状态下的滚动升级(rolling update)

■ 支持Job高速运行的各种机能

- 稳定高速的共享文件系统
- 可以最大化发挥系统性能的HPC扩张技能, Compiler, MPI执行环境等

富岳的Software Stack



性能解决方案的介绍

提高性能

- 提高Job的大页（hugepage）使用率
- 提升进程间通信性能

提高性能稳定性

- 防止内存碎片的生成
- 抑制系统噪音

■ 使用大页为什么可以提升性能

- TLB(Translation Lookaside Buffer)是存储虚拟地址和其映射的物理地址(pte)的缓存。当对虚拟地址进行读写操作时,如果pte不存在tlb中(tlb miss),为了取得保存在内存中的pte,就会产生内存访问。如果内存访问频繁发生,就会降低Job的性能。
- A64FX的tlb缓存可以存储1000个虚拟地址和物理地址的映射。
 - 2MiB大页可以保证对2GB(2MiB*1000)以内内存的读写不会发生tlb miss。
 - 64KiB (normal page)的普通页只可以保证对64MB(64KiB*1000)以内的内存读写不会发生tlb miss。
 - 在A64FX上面对2GiB内存进行随机的写操作,使用2MiB大页的性能约是64KiB普通页的2.2倍。

■ 分配大页的方式

- hugetlbfs
 - 需要用户自己调用mmap(2), 并且对flag参数指定MAP_HUGETLB来实现
 - 支持contiguous bit
 - ARM体系结构, normal page是64KiB的情况下, 支持的大页有2MiB(contiguous bit), 512MiB, 16GiB(contiguous bit), 4TiB
- Transparent HugePage (THP)
 - 用户自己不需要考虑大页, 调用malloc(3)内核会自动分配大页给用户。但是分配多少大页由内核决定(虚拟地址是2MiB对齐(alignment)并且申请分配的size是2MiB的倍数)
 - 不支持contiguous bit
 - ARM体系结构, normal page是64KiB的情况下, 支持的大页有512MiB, 4TiB

为何hugetlbfs支持2MiB的大页

- 不同的granule size = (normal page size) 所支持的大页 (granule size 在kernel编译时由kernel config决定。RHEL为64KiB)

granule size	page table段数	PAGE_SHIFT	PMD_SHIFT	PUD_SHIFT	PGDIR_SHIFT
4KiB	4	12	21	30	39
64KiB	3	16	29	42	-



granule size	page table段数	Available page size			
4KiB	4	4KiB(=2 ¹²)	2MiB(=2 ²¹)	1GiB(=2 ³⁰)	-
64KiB	3	64KiB(=2 ¹⁶)	512MiB(=2 ²⁹)	4TiB(=2 ⁴²)	-

- ARM体系结构的Contiguous bit机能

- 把左右相邻的页结合成一个大页

granule size		可结合页数					
4KiB		16					
64KiB		32					

granule size	page table段数	Available page size					
4KiB	4	4KiB	64KiB(=4KiB × 16)	2MiB	32MiB(=2MiB × 16)	1GiB	-
64KiB	3	64KiB	2MiB(=64KiB × 32)	512MiB	16GiB(=512MiB × 32)	4TiB	-

※红色是在可以利用contiguous bit机能的条件下可以使用的大页

■ 课题

- 标准的glibc库提供的分配内存的函数，因为只能使用THP的方式来获取大页，所以不能最大限度，最大效率的分配到大页

■ 解决方案

- 提供一个hugepage库，这个库置换glibc里面分配内存的函数(malloc等)。hugepage库实现的malloc通过利用mmap(2)，以hugetlbfs的方式获得大页。

■ 结果验证

- 链接hugepage库，并用malloc(3)分配20GiB的内存。通过确认获得的大页数量(/sys/kernel/mm/hugepages/hugepages-2048kB/surplus_hugepages)来进行结果验证

→結果 一共获得10264枚大页。也就是说分配的20GiB内存全都是大页。

提高(单一节点)进程间通信性能

■ 进程间通信的重要性

- 并行计算中MPI (Message Passing Interface) 被广泛利用。一般情况下一个核 (core) 对应一个MPI进程。
- 随着处理器的核数越来越多, 在一个处理器上面会存在越来越多的MPI进程, 所以说单一节点内的进程间通信的性能也就越来越重要

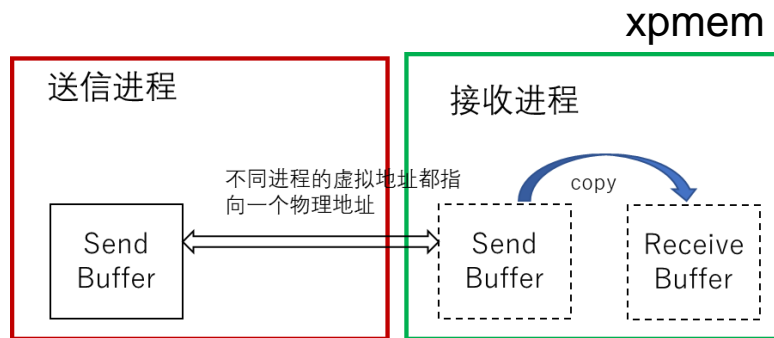
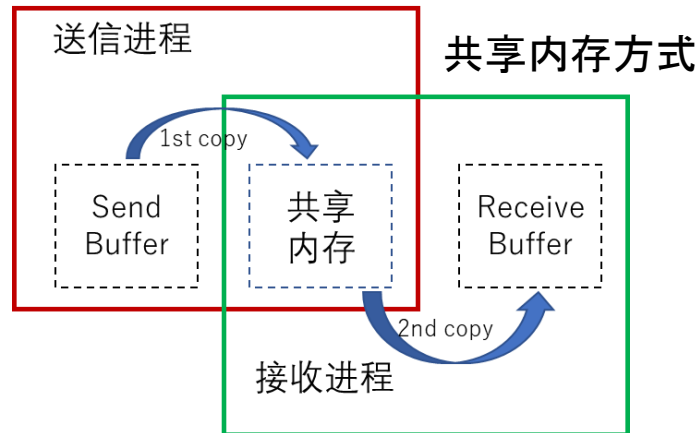
■ 典型的进程间通信

- 共享内存方式、pipe、FIFO、socket、CMA (Cross Memory Attach)
- RHEL里面自带的功能
- 一般需要进行两次copy

■ Xpmem

- 不是RHEL自带的功能, 是OSS的内核模块 (Kernel Module)
- 面向用户的接口简单
- 只需要一次copy

接收进程在进行缺页处理 (page fault) 的时候, 通过把虚拟地址指向送信进程的物理地址来实现进程间通信。显然这种方法比典型的进程间通信所要进行的内存copy的次数要少, 所以可以期待其通信性能的提高



■ 执行环境

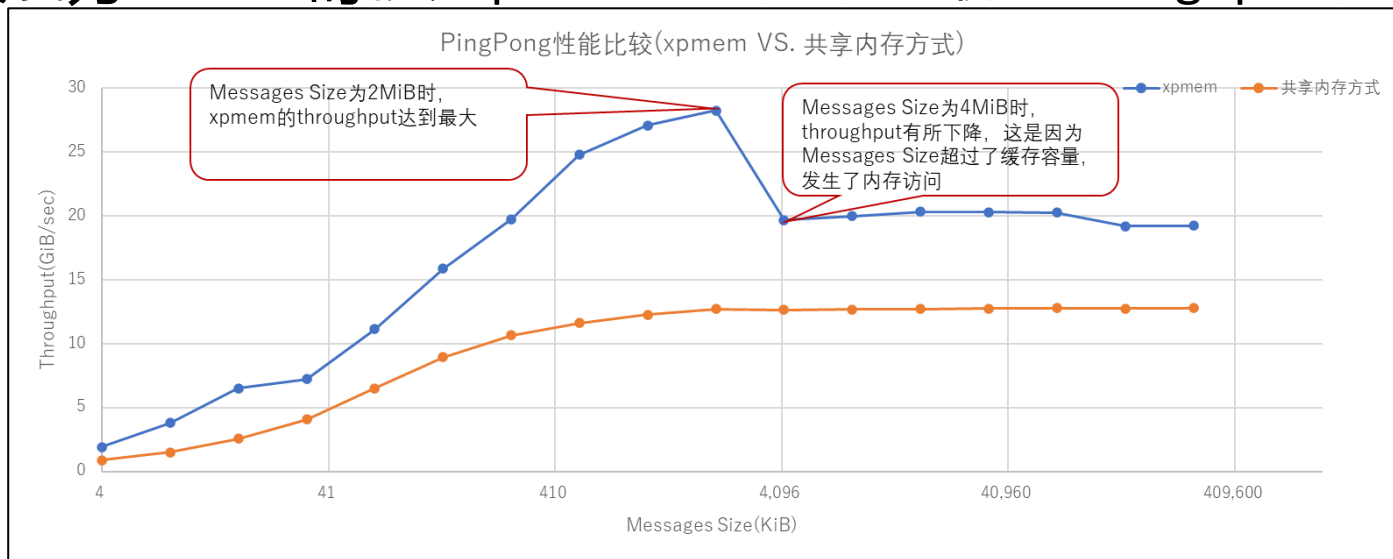
- Hardware : A64FX
- OS : RHEL8.3
- Middleware : TCS V4.0L20 (富士通的系统软件的名称)
- Benchmark : Intel MPI Benchmarks (IMB) 2018 Update 1

■ 测试项目テスト項目

- PingPong : 一对一通信
- Allreduce : 多对多通信

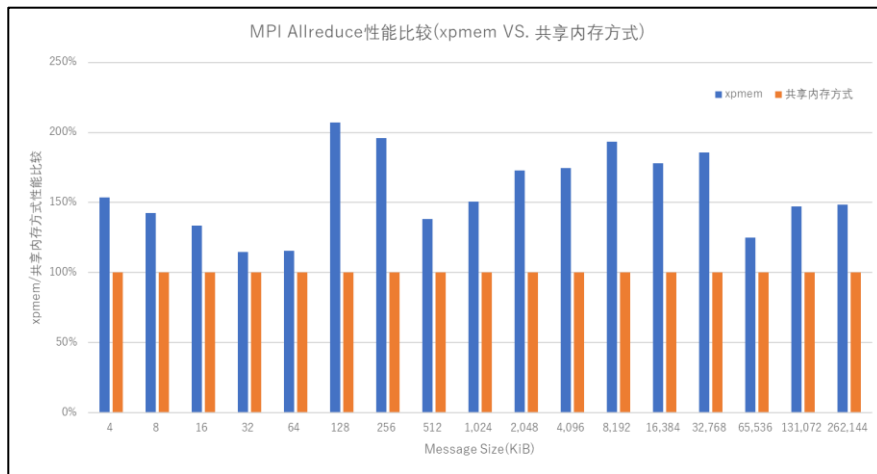
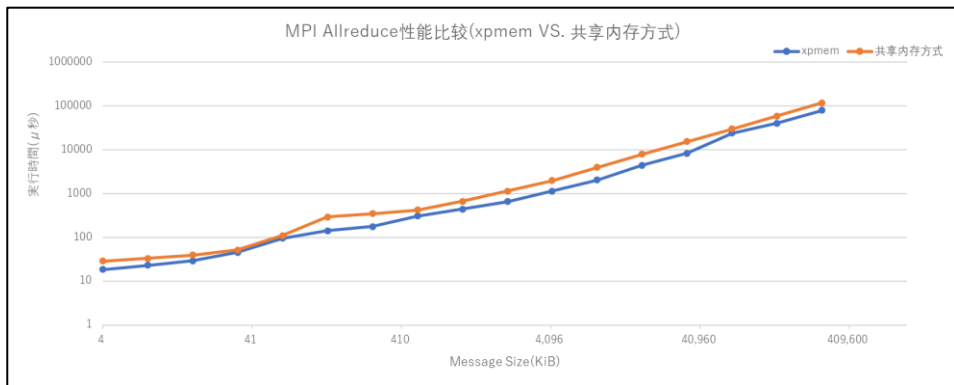
共享内存方式和xpmem方式的通信性能比较 (PiongPong)

- 对每个通信长度 (Messages Size) 进行1000次测试。通过计算每秒进行通信的量 (Throughput) 来进行性能比较
- 对所有通信长度, xpmem方式的性能都要超过共享内存方式。其中通信长度为2MiB的时候, xpmem方式达成了最大Throughput (2.8GiB/sec)



共享内存方式和xpmem方式的通信性能比较 (Allreduce)

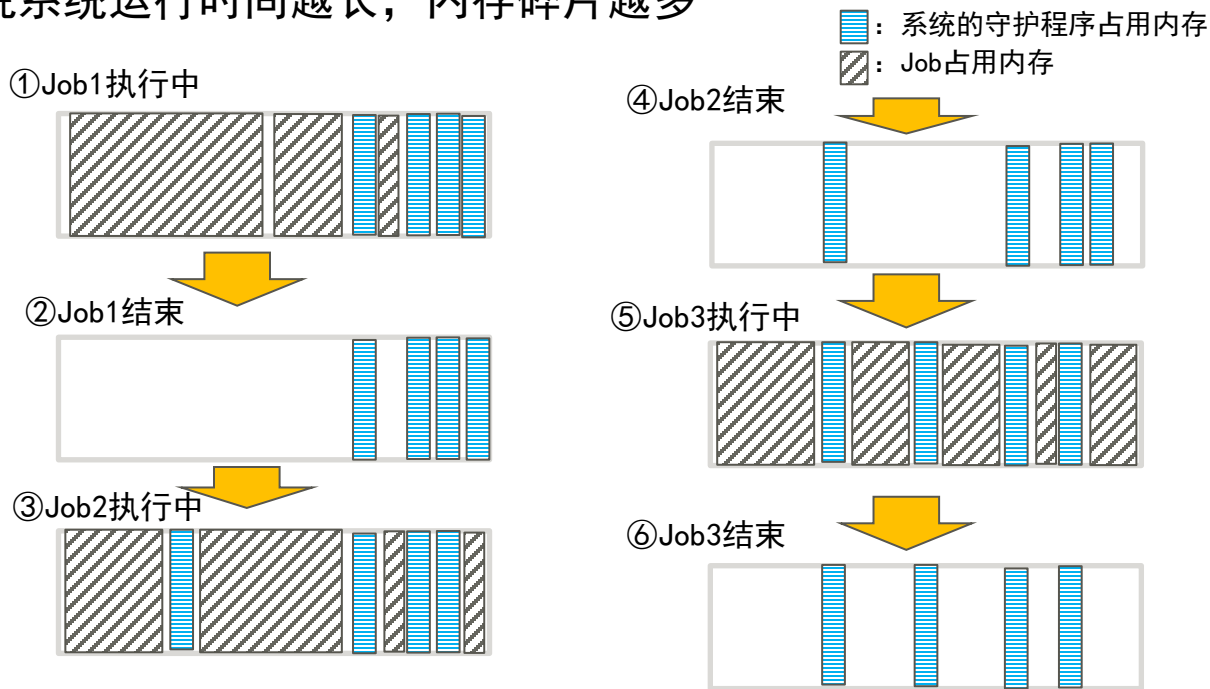
- 对每个通信长度 (Messages Size) 进行1000次测试。通过计算平均执行时间来进行性能比较。
- 对所有通信长度, xpmem方式的性能都要超过共享内存方式。



防止内存碎片的生成

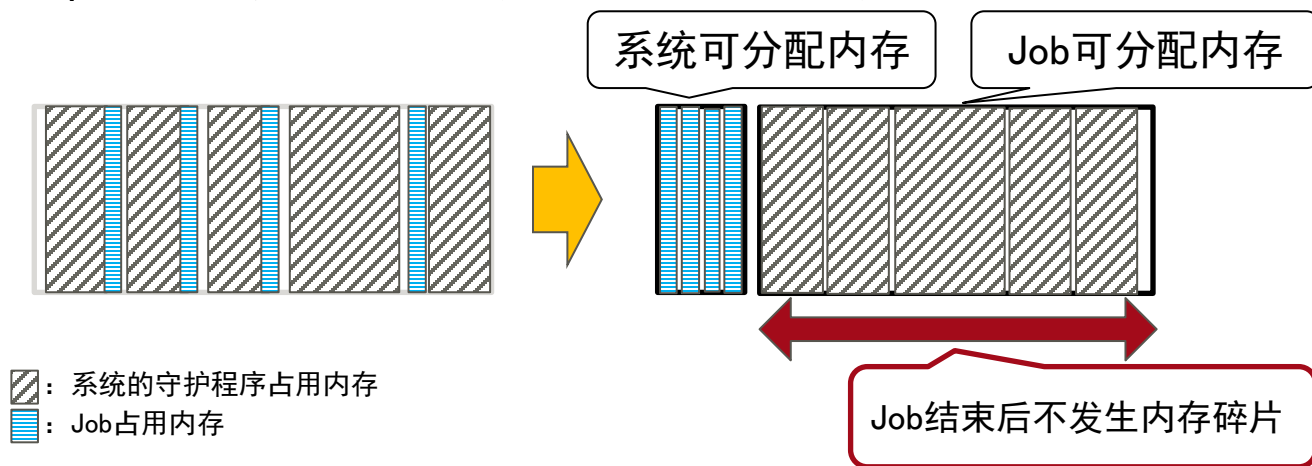
■ 课题：内存碎片的生成，会使大页减少，从而使大页的分配变得困难，影响Job性能

- 在Job运行的同时，系统的守护程序所分配的内存不会在Job结束后释放
- 一般来说系统运行时间越长，内存碎片越多



防止内存碎片的生成

- 解决方案：利用虚拟NUMA技术+cgroup机能，分离Job可以使用的内存和系统可以使用的内存。
 - 根据富岳设计，Job可分配内存允许占到内存总量的大约90%。单纯分离物理NUMA的话会使Job可以使用的资源减少，所以我们采用修改NUMA构成，构建虚拟NUMA的方法
 - 利用cgroup绑定系统使用的内存和Job使用的内存



防止内存碎片的生成 具体实现

■ 虚拟NUMA的生成

- A64FX共有4个物理NUMA，把它分割成8个虚拟NUMA。
- 通过cpufw来修改ACPI Table的SRAT (描述各个NUMA Node的构成要素)/SLIT (描述各个NUMA Node之间访问内存的性能) 达到分割NUMA的目的

关闭虚拟NUMA的numactl -H

```
node 0 cpus: 0 12 13 14 15 16 17 18 19 20 21 22 23
node 0 size: 8068 MB
node 1 cpus: 1 24 25 26 27 28 29 30 31 32 33 34 35
node 1 size: 8174 MB
node 2 cpus: 36 37 38 39 40 41 42 43 44 45 46 47
node 2 size: 8174 MB
node 3 cpus: 48 49 50 51 52 53 54 55 56 57 58 59
node 3 size: 8162 MB
```

开启虚拟NUMA的numactl -H

```
node 0 cpus: 0
node 0 size: 595 MB
node 1 cpus: 1
node 1 size: 637 MB
node 2 cpus:
node 2 size: 637 MB
node 3 cpus:
node 3 size: 631 MB
```

系统可分配内存

```
node 4 cpus: 12 13 14 15 16 17 18 19 20 21 22 23
node 4 size: 7473 MB
node 5 cpus: 24 25 26 27 28 29 30 31 32 33 34 35
node 5 size: 7537 MB
node 6 cpus: 36 37 38 39 40 41 42 43 44 45 46 47
node 6 size: 7537 MB
node 7 cpus: 48 49 50 51 52 53 54 55 56 57 58 59
node 7 size: 7531 MB
```

Job可分配内存

■ 利用cgroup分割内存

- /sys/fs/cgroup/cpuset下分别建立供系统使用和Job使用的slice
- system.slice/cpuset.mems : 0-3
- jobs.slice/cpuset.mems: 4-7

防止内存碎片的生成 结果验证

■ 测试环境

- Intel (R) Xeon (R) CPU E5520

■ 测试方法

- 反复执行模拟系统守护程序，保证系统用内存的使用率达到95%
- 连续执行Job。保证Job用内存使用率达到95%
- 12小时后，查看物理内存的状态

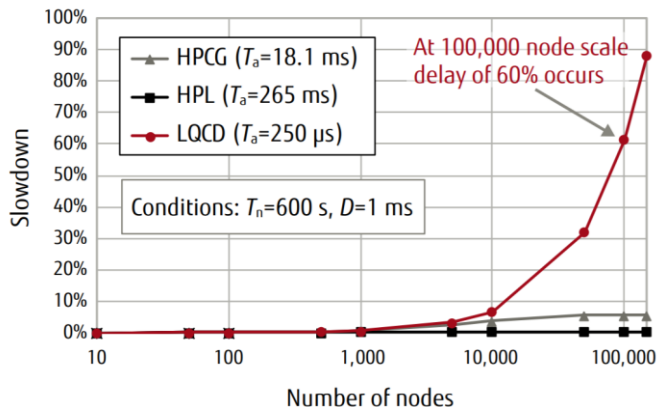
■ 测试结果

- 256MB以上物理连续的内存的比率从57%上升到88%



■ 前提知识

- 跟Job执行不相关的系统守护进程，内核线程，中断等会抢占Job的cpu资源，从而引起性能不稳定和性能的降低。系统的规模越大系统噪音带来的影响越明显
- 10万台计算集群组成的系统，由系统噪音带来的性能低下可高达60%



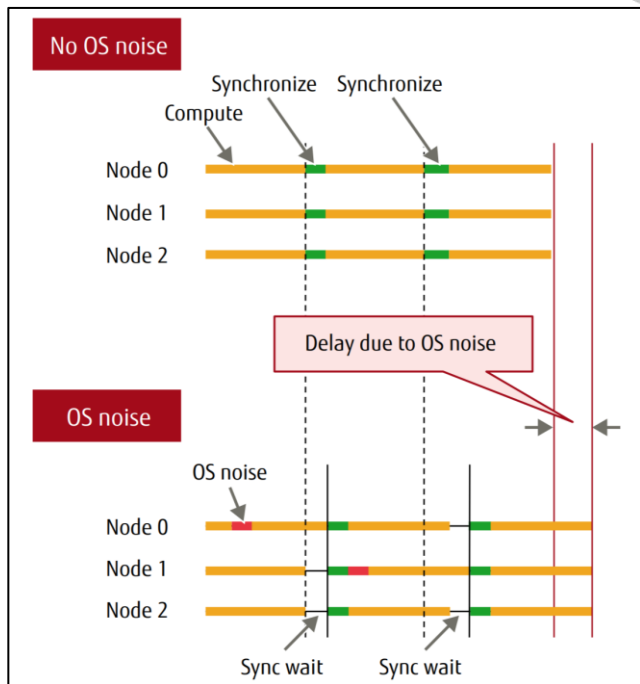
$$\text{Slowdown} = \left(1 - \left(1 - \frac{T_a}{T_n} \right)^N \right) \times \frac{D}{T_a}$$

D : Maximum noise length

T_n : Noise generation interval

T_a : Application synchronization interval

N : Number of nodes



■ 解决方案

- 利用cgroup把OS运行所必须的处理(系统守护进程, 内核线程, 中断)交给 Assistant Core
- 在内核的启动参数中添加nohz_full参数来减少时钟中断的次数
 - 如果设置了nohz_full参数那么在各个cpu (core) 的runqueue里面如果没有或者仅有一个任务(task)的情况下, 时钟中断的频率从CONFIG_HZ回/秒降到0回/秒
 - cat /proc/cmdline
nohz_full=12-59

开启虚拟NUMA的numactl -H

```
node 0 cpus: 0
node 0 size: 595 MB
node 1 cpus: 1
node 1 size: 637 MB
node 2 cpus:
node 2 size: 637 MB
node 3 cpus:
node 3 size: 631 MB
node 4 cpus: 12 13 14 15 16 17 18 19 20 21 22 23
node 4 size: 7473 MB
node 5 cpus: 24 25 26 27 28 29 30 31 32 33 34 35
node 5 size: 7537 MB
node 6 cpus: 36 37 38 39 40 41 42 43 44 45 46 47
node 6 size: 7537 MB
node 7 cpus: 48 49 50 51 52 53 54 55 56 57 58 59
node 7 size: 7531 MB
```

Assistant Core

系统可分配内存

Job可分配内存

抑制系统噪音 结果验证

■ 执行环境

- Hardware : A64FX
- OS : RHEL8.3
- Middleware : TCS V4.0L20 (富士通的系统软件的名称)
- Benchmark : FWQ (Fixed Work Quanta)
- 规模 : 1node

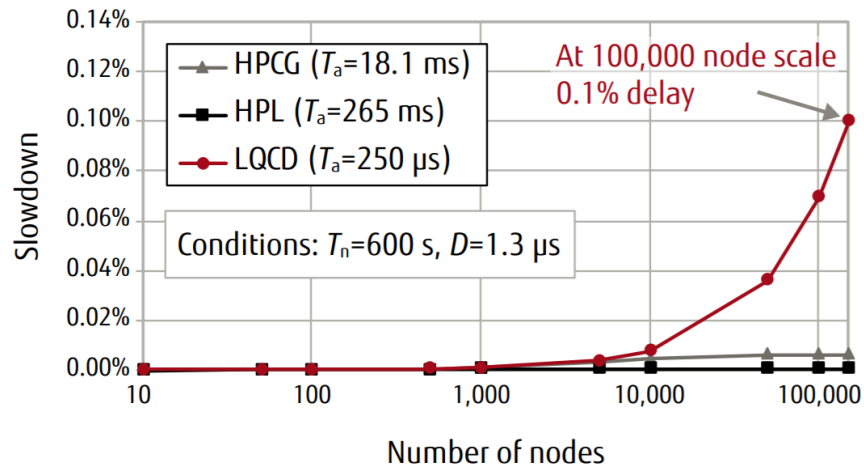
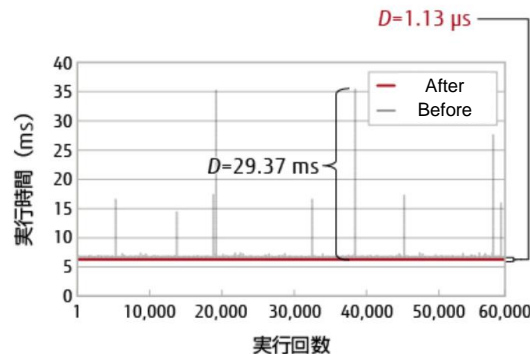
■ 実行結果

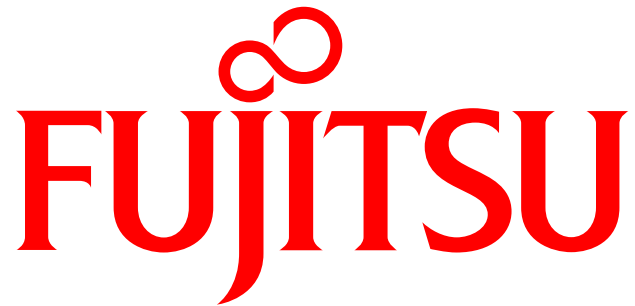
	最大噪音长度
「富岳」	1.13 μ s
「京」(参考)	85 μ s

■ 富岳规模的系统噪音预测

- 10万计算集群规模可以把系统噪音带来的性能低下降到0.1%
- 在富岳上面运行的结果将在SC21 (Supercomputer Conference) 21上发表

Linux vs. Lightweight Multi-kernels for High-Performance Computing:
Experiences at Pre-Exascale





shaping tomorrow with you