スーパーコンピュータ「富岳」におけるOSの強化機能

張　雷　　岡本　高幸　　石井　周一郎　　平井　浩一　　住元　真司　　Balazs Gerofi
髙木　将通　　石川　裕

あらまし

理化学研究所と富士通は，スーパーコンピュータ「京」（以下，「京」）の後継機として，2021年度の一般共用開始に向けてスーパーコンピュータ「富岳」（以下，「富岳」）の開発に取り組んでいる。「富岳」では「京」のソフトウェア資産を継続しつつ，計算性能・資源の利用効率や使い勝手など，様々な点について改良・改善に取り組んでいる。

本稿では，「富岳」のOSについて，独自OSではなく標準OSを採用するに至った経緯，OSレベルでの高性能化に対する取り組み，およびサポートするジョブ実行環境の拡大について述べる。

1. まえがき

理化学研究所と富士通は，スーパーコンピュータ「京」（以下，「京」）の後継機であり，2021年度の一般共用開始を予定しているスーパーコンピュータ「富岳」（以下，「富岳」）の開発に取り組んでいる。「富岳」では，「京」のソフトウェア資産を継続しつつ，計算性能･資源の利用効率や使い勝手など，様々な点について改良・改善に取り組んでいる。

「富岳」では，性能と運用性を両立するために，OSS（Open Source Software）コミュニティやOS（Operation System）ベンダー，および理化学研究所と協力し，OSの開発を行った。このOSは，汎用のLinuxディストリビューションに対して，「京」の開発で得られた性能チューニング技術と専用ドライバ開発の知見を適用したものであり，「富岳」が目指す汎用性と高性能の両立を実現している。

また「富岳」では，理化学研究所で開発しているHPC（High-Performance Computing）向けの軽量OSである，McKernelを使用したジョブも実行可能である。更に，Dockerを利用したコンテナジョブ，Linux Kernel仮想化技術を活用したKVM（Kernel-based Virtual Machine）ジョブも，エンドユーザーの用途に応じて選択して利用可能としている。

本稿では，独自OSではなく標準OSを採用するに至った経緯，OSレベルでの高性能化に対する取り組み，サポートするジョブ実行環境の拡大について述べる。

2. 標準ディストリビューションの採用

「京」で採用されたSPARCアーキテクチャーのCPUでは，OSSを含めたソフトウェアのエコシステムが活発ではなかったため，独自OSを採用した。そのため，OS版数は固定であり，利用可能なソフトウェアも限られていた。

これに対して「富岳」では，多数のベンダーが参画し活発に開発が進んでいるArmアーキテクチャーを採用することで，エコシステムの一員として，共に業界を発展させていくことを目指している。Armアーキテクチャーを採用することで，OSとして多くのLinuxディストリビューションが利用可能となったが，その中でも長期サポートの観点から，サーバ向けLinuxディストリビューションの一つであるRHEL（Red Hat Enterprise Linux）を採用することとした。これによって，独自OSでは困難であった定期的なバージョンアップやセキュリティパッチによる安全性の維持が可能となる。また，OSディストリビューション本体に限らず，OSSやISV（Independent Software Vendor）を含めた多数のソフトウェアの利用が可能になっており，「富岳」の目指す汎用性の確保に貢献している。

3. OSレベルでの高性能化に対する取り組み：OSノイズの極小化

「富岳」では，OSノイズの削減によって大規模並列実行時のアプリケーション性能を向上させている。本章では，OSノイズによる性能劣化の仕組みと，アシスタントコアを活用したノイズ削減手法について述べる。更に，HPC向けのLWK（Light Weight Kernel）であるMcKernelについて紹介し，そのノイズ削減効果について述べる。

3.1 OSノイズとは

スーパーコンピュータの価値は，利用者の計算アプリケーションを高速に実行することである。一方で，コンピュータシステムとして正しく動作させるためには，多くの管理処理も必要であり，それを実施している基盤ソフトウェアがOSである。

大規模なスーパーコンピュータで計算アプリケーションを実行する場合には，このOS管理処理によって，アプリケーションの実行が遅延して，性能向上の阻害要因となる場合がある。OSデーモンやカーネルデーモン，割り込み処理など，アプリケーション以外の処理によるアプリケーション実行遅延を総称して，OSノイズと呼ばれる。

図-1に，OSノイズによってアプリケーションの実行が遅延する様子を示す。現代のHPCアプリケーションは，複数ノードで実行することが基本であるが，複数ノードで連携して計算を行う場合，ノード間でデータをやり取りするために次の処理を待ち

合わせることが多い。この待ち合わせのことを，同期処理と呼ぶ。あるノードでOSノイズが発生すると，そのノードのアプリケーションの実行が遅延するが，更に同期処理を行うことによって，他のノードにまで遅延が伝搬することになる。このため，ノード数が多く大規模なシステムになるほど，小さなOSノイズでも大きな遅延につながる。

図-2に，OSノイズ対策を実施していない環境を想定したパラメーターにおける，ノード規模とノイズによる遅延の関係を示す。Slowdown（実行遅延）のモデル式は，文献[1]を参考にして導出した。スーパーコンピュータでよく利用される二つのベンチマーク（HPCG, HPL），および「富岳」のターゲットアプリケーションの一つであるLQCDについて，OSノイズによる性能劣化の度合いをモデル式により算出した結果である。アプリケーションの遅延に関係する要素は，最大ノイズ長（D），ノイズ発生間隔（T_n），アプリケーション同期間隔（T_a）である。D，T_n，T_aが一定とすると，ノード数の増加に伴って遅延も増大する。10万ノード規模を超えると性能が60％以上劣化するアプリケーション（D=1 ms，T_n=600 s，T_a=250 μs）も存在する。そのため，15万ノードを超える規模の「富岳」では，OSノイズ対策は必須である。

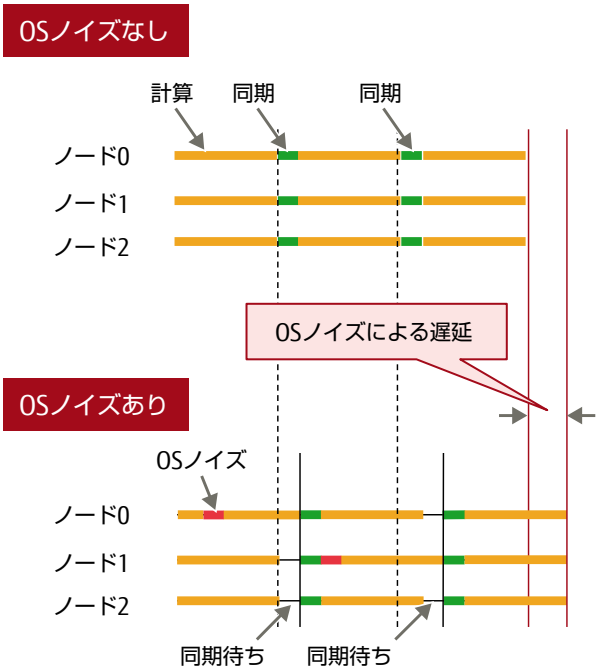
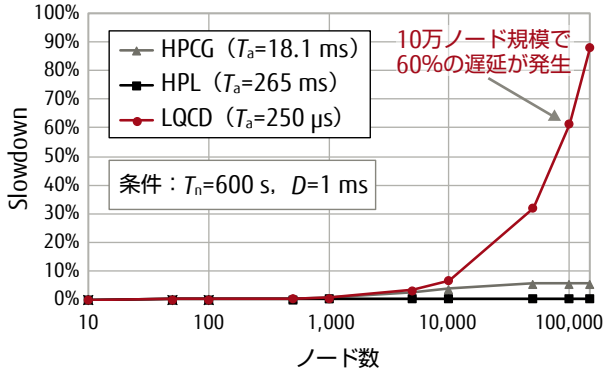


図-1　OSノイズによる遅延

「富岳」では，極限まで性能を追求するために，細かい粒度でノード間の同期を要求するアプリケーションに対して，チューニングの効果を見極めることが求められる。そのためには，局所的な性能ブレを最小限に抑制することが必要である。そのターゲットとして，全ノード規模でアプリケーション同期間隔T_a＝1 msのアプリケーション実行時に1％以下の性能ブレに抑えることを目指して，最大ノイズ長10 μsというチャレンジングな目標値を設定し，開発を行った。

3. 2　OSノイズ対策1：アシスタントコアの活用および効果

「富岳」では，OSノイズの発生源になるシステムデーモン，カーネルデーモン，割り込み処理といったOSタスクを実行するためのCPU資源を，アプリ



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

D：最大ノイズ長
T_n：ノイズ発生間隔
T_a：アプリケーション同期間隔

図-2　ノード規模とノイズによる遅延の関係

ケーションを実行するためのCPU資源と分離することで，OSノイズの影響軽減を図っている。

「富岳」では，計算ノードには1ノードあたり50コアが，IO兼計算ノードには1ノードあたり52コアが搭載されている。そのうち48コアはアプリケーション計算用として割り当てられ，計算コアと呼ばれる。残りの2コアまたは4コアは，システム用としてOSタスクに割り当てられ，アシスタントコアと呼ばれる。なお，OSタスクのコアへの割り当て（バインド）は，Linuxカーネルのcgroup機能を使って実装している。

図-3に，OSタスクのアシスタントコアへのバインドを有効および無効にした場合の，OSノイズ測定結果を示す。アプリケーションは，FWQ（Fixed Work Quanta）[2] を使っている。FWQは，一定の処理を繰り返し実行し，OSノイズの大きさを測定するベンチマークである。OSノイズの大きさは，ノイズ長と呼ばれる。ノイズ長は，各回の実行時間と一番短い実行時間との差分である。グラフの横軸は，測定の実行回数を示す。また，縦軸は処理の実行時間を示しており，これがまれに増加するのがOSノイズによる遅延である。バインドを無効にした場合，最大実行時間は35.504 ms，最大ノイズ長D=29.37 msになっており，実行時間のばらつきが大きいことが観測できた。一方で，バインドを有効にした場合，6万回の測定を行っても，最大実行時間6.129 ms，最大ノイズ長D=1.13 μsであった。この結果から，計算コアにはアプリケーションを，アシスタントコアにはOSノイズの発生源になるOSタスクを，それぞれ分離してバインドすることで，OSノイズの影響が軽減されていることが確認できた。なお，測定に利用したOSは，RHEL8.2GA（RHSA-2020:2427）である。

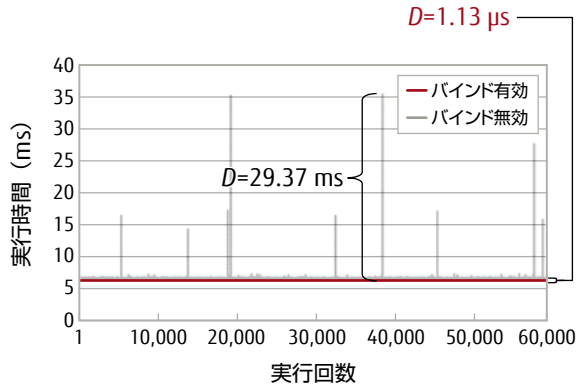


図-3　アシスタントコアへのバインド有無によるOSノイズ比較

前述のように，RHELのような汎用的なサーバ向けOSを利用しつつ，OSノイズの低減を目指しているが，それでも完全にOSノイズを取り除くことはできない。これは，Linuxでは様々なユースケースを対応するために，アシスタントコアにバインドできないカーネルスレッド（例えば，kswapd，kworker）が存在するためである。

このようなLinux実装に固有で避けることのできないOSノイズは，Linux以外の軽量OS（LWK）を採用することによって削減可能である。これは，大規模環境における性能を重視するHPCアプリケーションに向いている。代表的なLWKとして，Argonne National Laboratoryの「Argo」[3]，Sandia National Laboratoriesの「Hobbes」[4]，Intelの「mOS」[5]，理化学研究所の「McKernel」[6] などがある。McKernelは当初，東京大学でスクラッチから開発され，その後理化学研究所が開発を継承したLWKである。「富岳」では，極限までOSノイズを削減するために，McKernelによるジョブ実行をサポートする。

McKernelの概念図を図-4に示す。McKernelは，Linuxとは独立したCPUとメモリーで動作し，HPCアプリケーションにおいて性能クリティカルとなる，プロセス管理，シグナル処理，メモリー管理，同期処理などのOS機能を，Linuxとは独立して実装している。これによって，効率の良いプロセス・メモリー管理が実現されている。McKernelが提供しないOS機能は，Linux側で実行される。OSノイズの要因となるデーモンや割り込み処理は，Linuxが動作しているOS Core（アシスタントコア）上で処理され，McKernelが動作しているApp Core（計算コア）には影響を与えない。HPCアプリケーションがMcKernelのOS機能のみ利用している場合には，Linuxに起因するOSノイズ影響を受けない。

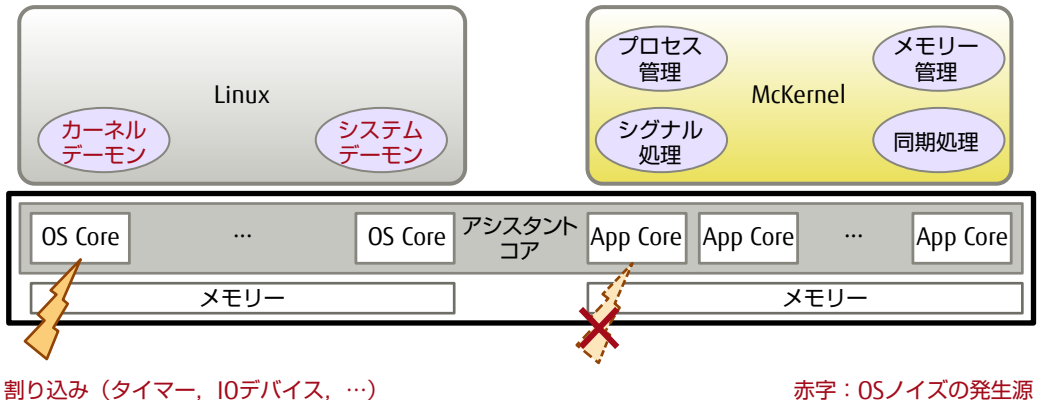


図-4　McKernelの概念図

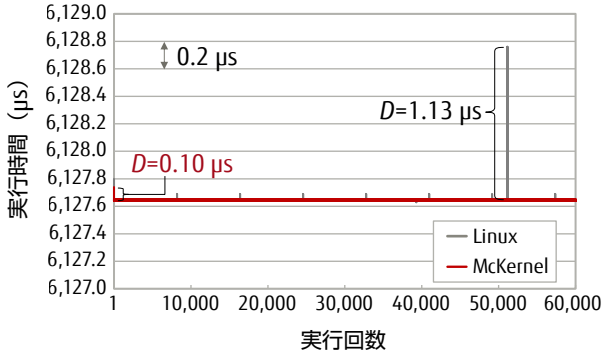


図-5　LinuxとMcKernelのOSノイズ比較

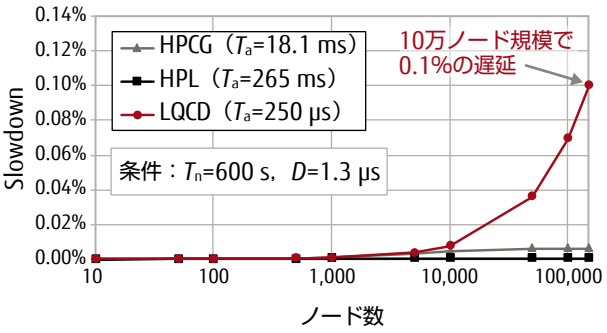


図-6　アシスタントコアバインドによる
OSノイズの削減効果

3.3　OSノイズ対策2：McKernelの採用および効果

図-5に，Linux上とMcKernel上で，OSノイズ測定プログラムFWQを使ってノイズ測定を行った結果を示す。図-3では縦軸の目盛りがミリ秒（ms）のオーダーであったのに対して，この図ではマイクロ秒（μs）のオーダーであることに注意されたい。OSタスクをアシスタントコアにバインドしたLinux上の測定結果は，図-3と同じデータを使用し，最大実行時間6.129 ms，最大ノイズ長D=1.13 μsであった。図-3のオーダーでは見えなかったLinuxOS上でのOSノイズも，このように非常に小さいオーダーでははっきりと見える。一方で，McKernel上では最大実行時間6.128 ms，最大ノイズ長D=0.10 μsであり，マイクロ秒のオーダーで見ても極めて小さいことが分かる。これによって，McKernel上ではLinuxに比べて更にOSノイズ影響が少ないことが確認できた。

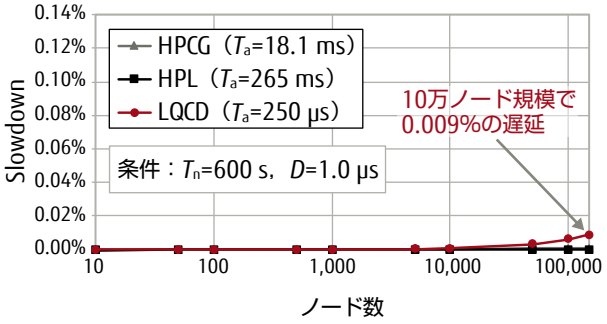


図-7　McKernelジョブ実行環境による
OSノイズの削減効果

3.4　性能劣化モデル式による全系システム評価

最後に，このOSノイズ対策の効果を，図-2で示した各ベンチマークに対する性能劣化シミュレーションで比較する。

図-6にアシスタントコアバインドを有効化した場合，図-7にMcKernelでジョブを実行した場合の，

OSノイズによる性能劣化の計算結果をそれぞれ示す。図-2で示したように，ノイズ対策を実施しなかった場合の最大遅延は60％であったのに対して，アシスタントコアへのバインドを有効とした場合では0.1％，McKernelでジョブ実行した場合では0.009％まで性能劣化を抑制することができるであろうという結果を得た。

4. 様々なジョブ実行環境のサポート

「富岳」では，多様なユーザーニーズに応えるために，アプリケーションの特徴や実行目的に応じて，以下に示すような複数のジョブ実行環境を用意している。更に，ユーザビリティ向上のために，様々なジョブ実行環境を同じユーザービューで利用することが可能になっている。

・通常モード

基本的なジョブ実行環境モードである。PCクラスタなど，他のHPCシステムとも互換性が高い。McKernelでは，性能劣化につながるIO関連処理やシステムコールの呼び出し回数が多いアプリケーション，あるいは特徴が把握されていないアプリケーションに対して，利用が推奨される。

・McKernelモード

性能を優先するユーザー向けのモードである。演算が主な処理で，IO関連処理などシステムコールの呼び出し回数が少ないアプリケーション実行時は，通常モードと比べて高い計算性能を持つ。

更に「富岳」では，これらのジョブ実行モードに加えて，仮想環境の実行モードもサポートする。具体的には，DockerモードとKVMモードを用意している。

・Dockerモード

ホストOS上に，Linuxコンテナと呼ばれるホストOSから隔離された環境を用意し，ホストOSと異なるOS（コンテナイメージ）上でプログラムを動作させることができる。

・KVMモード

Linuxのハードウェアの仮想化支援を利用することで，仮想化環境上においてジョブを実行するモード。カーネルモジュールなどのOSカーネル層を開発するシステム開発者のように，プログラム実行に特権を必要とするユーザーに対して有効である。

DockerモードおよびKVMモードについては，システム管理者が用意する環境の他，ユーザーが独自に用意した環境も利用することができる。ジョブ実行の方法に変わりはなく，例えば，以下に示すようなジョブ実行を指示するコマンドラインにおいて，jobenvオプションでジョブ実行モードを指定するだけで，モードを切り替えることが可能である。

・通常モード

```
#pjsub -L u,rg=rg runA.sh
```

・McKernelモード

```
#pjsub -L rg=rg,jobenv=mck runA.sh
```

・Dockerモード

```
#pjsub -L rg=rg,jobenv=docker runA.sh
```

・KVMモード

```
#pjsub -L rg=rg,jobenv=kvm runA.sh
```

5. むすび

本稿では，「富岳」のOSについて，OSレベルでの高性能化に対する取り組みと，多様なユーザーニーズに応えるための様々なジョブ実行環境への対応について説明した。

OSレベルでの高性能化に対する取り組みにより，15万ノード以上の「富岳」規模でも性能劣化を0.009％以下に抑え，アプリケーションの高速実行の土台を整えることができた。今後は，LinuxでもOSノイズの更なる削減を目指していきたい。

また，ジョブ実行環境については，最新のコンテナ，KVMと同様にMcKernelもユーザーが独自に用意したMcKernelの実行をサポートするなど，ユーザー利便性の向上に貢献していきたい。

参考文献・注記

[1] D. Tsafrir et al.：System noise, OS clock ticks, and fine-grained parallel applications. 19th annual international conference on Supercomputing, 2005.

[2] Lawrence Livermore National Laboratory：FTQ/FWQ summary (v1.1). https://asc.llnl.gov/sites/asc/files/2020-06/FTQFTW_Summary_v1.1.pdf

［3］ Argonne National Laboratory：“Argo: Low-Level Resource Management for the OS and Runtime.” https://www.anl.gov/mcs/argo-lowlevel-resource-management-for-the-os-and-runtime
［4］ Sandia National Laboratories：Hobbes - An Operating System for Extreme-Scale Systems. https://xstack.sandia.gov/hobbes/
［5］ GitHub：What is mOS for HPC. https://github.com/intel/mOS/wiki
［6］ IHK/McKernel：IHK/McKernel Documentation. https://ihkmckernel.readthedocs.io/en/latest/

著者紹介

張　雷（ちょう　らい）
富士通株式会社
プラットフォームソフトウェア事業本部
Linux開発に従事。

岡本　高幸（おかもと　たかゆき）
富士通株式会社
プラットフォームソフトウェア事業本部
Linux開発に従事。

石井　周一郎（いしい　しゅういちろう）
富士通株式会社
プラットフォームソフトウェア事業本部
Linux開発に従事。

平井　浩一（ひらい　こういち）
富士通株式会社
プラットフォームソフトウェア事業本部
「富岳」の運用系ソフトウェア開発に従事。

住元　真司（すみもと　しんじ）
富士通株式会社
プラットフォームソフトウェア事業本部
「富岳」の運用系ソフトウェア開発に従事。

Balazs Gerofi
国立研究開発法人理化学研究所
計算科学研究センター
フラッグシップ2020プロジェクト
システムソフトウェア開発チーム
McKernelの開発に従事。

髙木　将通（たかぎ　まさみち）
国立研究開発法人理化学研究所
計算科学研究センター
フラッグシップ2020プロジェクト
システムソフトウェア開発チーム
McKernelの開発に従事。

石川　裕（いしかわ　ゆたか）
国立研究開発法人理化学研究所
計算科学研究センター
フラッグシップ2020プロジェクトを統括。