「京」におけるジョブ I/O によるローカル MDS 負荷の分析

黒田 明義 [1)], 井上 俊介 [2)], 関澤 龍一 [3)], 南 一生 [1)]

1) 理化学研究所 計算科学研究機構
2) 株式会社富士通 解析ソリューション部
3) 富士通株式会社 テクニカルコンピューティングソリューション事業本部

kro@riken.jp

Analysis of the Load of Local Meta Data Server Access from the Job I/O on the K computer

Akiyoshi Kuroda [1)], Shunsuke Inoue [2)], Ryuichi Sekizawa [3)], Kazuo Minami [1)]

1) Advanced Institute for Computational Science, RIKEN
2) Analysis Solution Department, FUJITSU, LTD.
3) Technical Computing Solutions Unit, FUJITSU, LTD.

概要

近年 HPC システムの運用の現場で，様々な問題が報告されている．ジョブ実行時間の変動の問題は, 要因が複雑に絡まり, 原因の究明や解決が難しい問題の一つである. 変動が大きいものの中には，ジョブが予定時間内に終了しない事例もあり，解析により，分散ファイルシステムでの I/O によるメタデータサーバ(MDS)の負荷が，性能限界まで達している事例があることが分かった．本報では，ノード単位で蓄積されている MDS 処理命令数のカウンタ情報を集約し，ジョブとの関係を解析するためのシステムを構築したので，その概要並びに解析事例を紹介する．本解析により，MDS 負荷が高いジョブの影響で，他のジョブが想定時間内に終了しない事例が見つかった．また小規模な並列数のジョブでも MDS 負荷が高いものがあることが分かった．MDS 負荷が高い状態では I/O 処理の一部が滞るため，過負荷状態の対策を行うことは，計算資源の無駄な消費の削減につながる．

1 はじめに

近年 HPC システムの運用の現場で，様々な問題が報告されている．ジョブ実行時間の変動の問題は，要因が複雑に絡まり，原因の究明や解決が難しい問題の一つである．変動が大きいものの中には，ジョブが予定時間内に終了しない事例もあり，解析により，分散ファイルシステムでの I/O によるメタデータサーバ（以下，MDS と記す）の負荷が，性能限界まで達している事例があることが分かった．本報では，スーパーコンピュータ「京」（以下，「京」と記す）上でノード単位で蓄積されている MDS 処理命令数のカウンタ情報を集約し，ジョブとの関係を解析するためのシステムを構築したので，その概要並びに解析事例を紹介する．本解析により，MDS 負荷が高いジョブの影響で，他のジョブが想定時間内に終了しない事例が見つかった．また小規模な並列数のジョブでも MDS 負荷が高いものがあることが分かった．MDS 負荷が高い状態では I/O 処理の一部が滞るため，過負荷状態の対策を行うことは，計算資源の無駄な消費の削減につながる．

第 2 章では，本研究の対象であるジョブ I/O に起因する MDS の負荷調査の背景並びに使用した「京」のシステムの特徴について紹介する．第 3 章では，解析のために構築したジョブ I/O による負荷状況のデータベースについて紹介し，解析手法を紹介する．第4章では，実際にMDS負荷が高いジョブの解析事例を紹介し，第 5 章でまとめと今後に向けた課題を議論する．

2 LMDS 負荷評価の背景

本章では，本研究の対象であるジョブ I/O に起因する MDS 負荷の評価の背景並びに, 評価に使用した「京」のシステムの特徴について紹介する．

2.1 「京」のファイルシステム

本節では，「京」のファイルシステム構成について説明する．図 1 に「京」のシステム構成を示

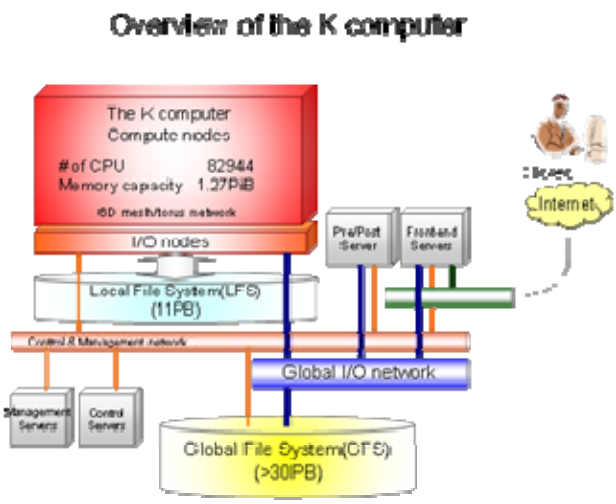

図 1 「京」のシステム概念図

Figure 1 Concept of the K computer system

す．「京」は 1 ラックあたり 96 台の計算ノードを格納し，システム全体では 864 ラック，82,944 台の計算ノードを有する．

計算ノードのファイル I/O 性能を確保することを目的に，プログラムが実行される際に使用するファイルシステム（Local File System：以下，LFS と記す）と，ユーザのプログラムやデータを格納するファイルシステム（Global File System：以下，GFS と記す）の 2 階層のファイルシステムで構成されている．「京」の LFS 並びに GFS には，富士通が Lustre バージョン 1.8 系をベースに独自の機能拡張を行った FEFS（Fujitsu Exabyte File System）[1]が使用されている．GFS は，可搬性を考慮して複数ボリュームで構成されているのに対し，LFS は全計算ノードから一様なアクセスを実現するために 1 つのボリューム構成になっており，MPI プログラムでランク間を共有してアクセスすることが可能な領域（共有ディレクトリ）が提供されている．

計算ノードとファイルシステムの間には，I/O 用のノード（以下，I/O ノードと記す．）により入出力やシステムファイルへのアクセスを行う．LFS と GFS へのアクセスに用いる I/O ノードとしてそれぞれ LIO および GIO がある．LFS と GFS の間は I/O ノードを介して InfiniBand および「京」のノード間インターコネクトである Tofu インターコネクト[2]により接続されている．ステージング機構[3][4]を用いて，GFS と LFS の間のファイル転送処理を行っており，この処理は他のジョブ実行と並行して行われるため，ジョブの実行効率向上に大きく貢献している．いずれの LFS のアクセス領域も OSS（Object Storage Server）として使用されている LIO を通してアクセスが可能である．また，LFS には，ファイル I/O の高速化のために，ランクごとに独立にアクセスできる領域（ランク番号ディレクトリ）が，loopback デバイスにより提供されている．

2.2 メタデータサーバ（MDS）

分散ファイルシステムでは，計算ノードなどにおける I/O に関する制御情報を管理する必要がある．管理が必要となるファイルシステムの制御情報には，空き領域，i ノード，間接エクステントブロック，ディレクトリ，シンボリックリンクなどの情報があり，これらをメタデータと呼ぶ．

各ノードで独自にこれらの処理を行うと，一様なアクセス性能が保証できなくなるため，大規模なファイルシステムでは，一元的に管理するサーバが用意されることが多い．メタデータ情報を管理するサーバのことをメタデータサーバ(MDS)と呼ぶ．MDS はファイルシステムのボリュームごとに稼働しており，LFS 上の MDS を LMDS と呼び，GFS 上の MDS を GMDS と呼ぶ．

計算ノード上で open/close などのファイル操作を行う際，計算ノードが MDS にメタデータを要求し，MDS はメタデータを返す．その後，各ノードから OSS に対し read/write などの I/O 処理が行われる．大規模な分散ファイルシステムでは，全てのノードで大量のファイルを作成したり(図 2 上)，共有領域のファイルを複数のノードからのアクセスする(図 2 下)ことで，MDS の負荷が高くなる．

LMDS が高負荷になると，LFS 内にあるファイルに対するメタアクセス処理が遅延する．LFS はジョブ実行に使用される領域である，このため，実行中のジョブは LMDS の応答が復旧するまで処理を待つことになり，遅延などの影響が発生する．また，計算ノードや GIO からのリクエストのうちの幾つかの処理は，LMDS からの応答が無い状態が一定時間続くと，各ノードは異常が発生したと判断し，強制停止する場合がある．但し，2.1 節で述べた loopback デバイスを利用すると，メタデータのアクセス要求の問い合わせ先を LFS 上の各 OSS に分散できるため，同様の処理でも LMDS の負荷を抑えることが可能である[5]．Loopback デバイスの作成についても LMDS の負荷への影響は

ない．作成にはLFS上のOSSが担当し．ランクごとに配置されたディレクトリ配下で作成処理が行われるためである．

GMDS が高負荷になった場合は，GFS 内にあるファイルに対するメタアクセス処理が遅延する．ステージングを行わないジョブで遅延が発生したり，ファイル操作に失敗する可能性がある．ただし，GFS はボリュームを複数に分割されており，ボリューム数分のGMDSが存在するため，影響範囲をボリューム単位に限定することが可能である．

「京」では負荷試験の結果から，MDS 処理の限界性能は 40,000[OPS]程度と見積もられている．これは，MDS の演算処理能力に起因するとされる．なお単位の[OPS]は，MDS が 1 秒間に処理した命令数である．また，限界性能の値はメタデータの処理内容によって変動すると想定される．

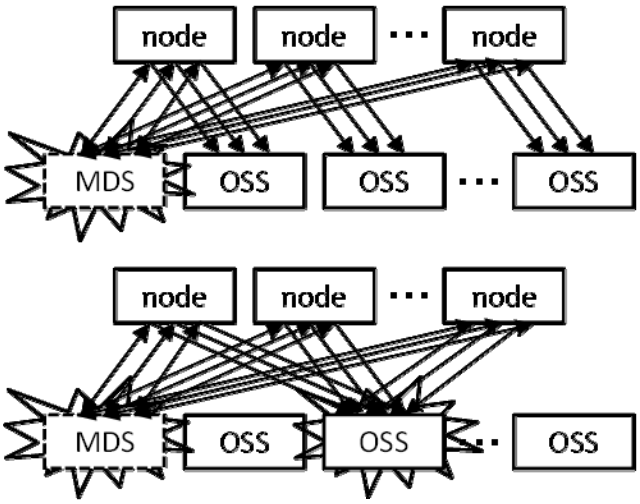

図 2　計算ノードからの MDS の負荷の概念図

Figure 2 Concept of MDS request from nodes

2.3 LMDS 負荷によるシステム停止事例

「京」運用開始後，I/O に関するトラブルが何件か寄せられてきた．トラブルの症状は多様であり，ユーザがファイルの編集に時間がかかるといったものから，ステージング処理の経過時間の超過，ジョブ実行時間の超過，最悪の場合システムの停止などがあり，当初は，原因の一つとして I/O のアクセス集中に伴うものと考え，ファイル I/O 量やファイル数に着目した調査や対策を行ってきた[6]．各種ログを解析すると，LMDS の処理能力が限界に達している事例が見られることが分かった，

図 3 は 2016 年 8 月に起こった障害の時の LMDS アクセス状況である．実線があるジョブの LMDS 処理命令数で，破線は他のジョブによる LMDS 処理命令数である．縦軸は 10 分間隔の LMDS 処理命令数で，1 秒あたりに換算すると 21,000[OPS]となり，システム性能の限界とされる 40,000[OPS]に迫る処理命令数である．この時は，GIO や計算ノードが停止するなどの問題が発生した．原因を調べると，LMDS の高負荷により，リクエスト処理のタイムアウトが発生し，GIO が停止．また，LFS の応答低下によるタイムアウトが発生し，計算ノードが停止したものと判明した．本件は，LFS 上の共有領域を用いて大量の I/O 処理を行っていることが分かり，ユーザが loopback デバイスの利用に切り替えることで対応できた．

このように LMDS の過負荷は自分のジョブの実行時間が延びるだけでなく，他のジョブやシステムへの影響も少なからずあると考えられ，ジョブ I/O に伴う LMDS の過負荷状態の調査が必要となった．

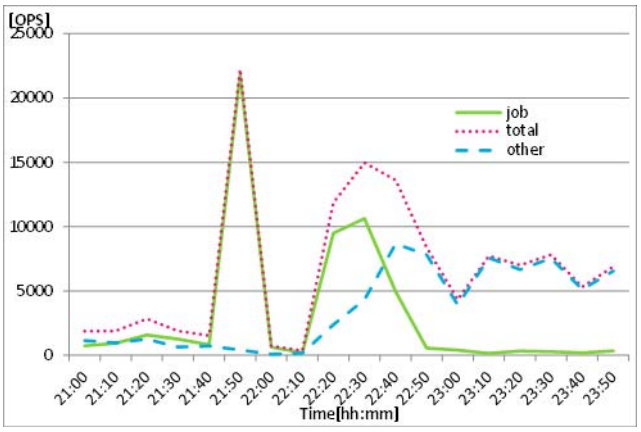

図 3　MDS 負荷の推移（2016/8）

Figure 3 Transition of MDS load (2016/8)

3　解析手法

本章では，ジョブの I/O の関する LMDS の負荷状況を解析するために構築したデータベースについて記載する．

3.1 LMDS データベースの構築

「京」では，運用効率改善を目指し，アプリケーションごとに性能改善に取り組んできた[7]．この取り組みをサポートするべく，ジョブ性能情報を自動的に採取し，データベースとして蓄積する

ジョブ性能情報解析システムの構築を行い，ユーザへの情報提供や運用効率改善のための解析に使用してきた．本システムは，様々なサーバ上に散在するジョブに関するデータをジョブ単位で集計し，一元管理するシステム（図 4）であり，I/O 関係では，全ノードで記録されている I/O 量情報から，I/O 量の合計値や最大値や平均値などを算出して蓄積している．

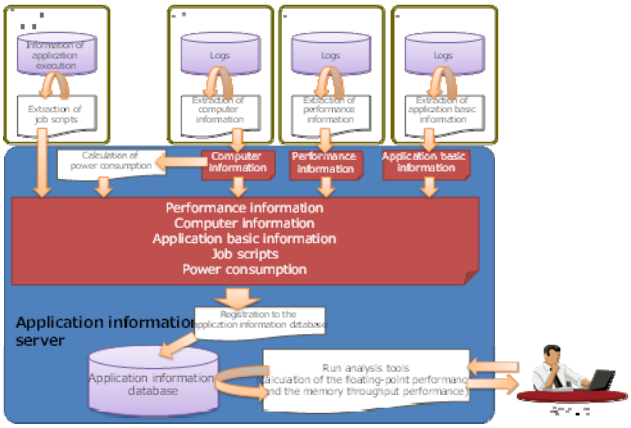

図 4　ジョブ性能情報解析システム概念図

Figure 4 Concept of the application information server system

3.2 ジョブ単位の LMDS 情報

今回，ジョブの I/O に伴う LMDS の負荷情報を，3.1 節で紹介したデータベースに追加する作業を実施した．LMDS の負荷情報は，常時ノード単位でアクセスの種類ごとの積算値が LMDS 内に蓄えられており，これらのデータをジョブ単位で集約し直す必要がある．

ジョブ性能情報解析システムでのデータベースに LMDS 負荷情報を蓄積するためには，いくつかの課題がある．LMDS 負荷情報はノード単位で積算値として蓄積されている．システム全体 82,944 ノードの情報を集約するには，LMDS 負荷はシステムの処理能力の制限により，約 2 分の時間を要することが分かった．このため，LMDS の負荷情報の蓄積のための時間方向の粒度を 10 分より細かくすることは困難であった．集約作業は，LMDS 内で行われるが，LMDS 内のボリュームは LFS のボリュームとは別であるためジョブのパフォーマンスに与える影響は小さいと想定している．

またジョブが使用する計算ノードの情報は，図 4 中の"application information server"と記載されているジョブ管理サーバで管理されている．このことは，ジョブが使用する計算ノードだけ，LMDS 負荷をリアルタイムに抽出するのは困難であることを意味する．

今回の LMDS 負荷情報のデータベース構築では，まず全ノードの LMDS 負荷情報を 1 度ジョブ情報解析システム上のサーバに集約し，その後，別途取り寄せたジョブが使用するノード情報を使用して，ジョブ単位の LMDS 負荷情報として加工し，蓄積を行った．これにより，他のジョブへの影響などの詳細な評価を，ジョブ単位の LMDS 負荷情報から行うことが可能である．なお，Lustre のバージョン 2.0 以降では，ジョブごとの MDS 負荷をコマンド実行レベルで取得可能である．

3.3 ジョブ単位の集計方法

ジョブ単位の LMDS 負荷情報に加工するにあたり，ジョブが開始・終了する時刻に必ずしもデータを抽出できるとは限らないという問題がある．3.2 節で記したように LMDS の時間方向の粒度は，全システムのノード情報の集約処理が律速となり，10 分間隔なっている．このためジョブの開始・終了時刻は，この精度での解析となる．また実際の情報集約には 2 分程度の時間がかかるため．一部のノード情報のみ収集される可能性も考えられる．図 5 にジョブ実行時間と LMDS 負荷情報のサンプリングのパターンを記す．図 5 中の「●印」で記載されているタイミングのみ採取する「ジョブ実行時間内だけのサンプリング」と図 5 中の「○印」と「●印」の両方のタイミングで採取する「ジョブ実行時間を含むサンプリグ」の 2 つの方法が考えられる．データベースにはこれら両パターンについて，全てのサンプリング点での LMDS 負荷情報並びに，サンプリング点数，最大値，平均値，最小値などの統計情報を蓄積している．

ジョブ I/O による LMDS 負荷の解析には，他のジョブの影響を排除するため．主にジョブ本体のみのサンプリングデータを使用する．実行時間が 20 分より短いジョブについては，サンプリング数が 0 となることがあるため．そのジョブを含む広い時間範囲のサンプリング情報を用いて解析を実施した．本 LMDS 負荷状況のデータ採取は 2016 年 1 月 15 日から開始しており，現在も蓄積している．

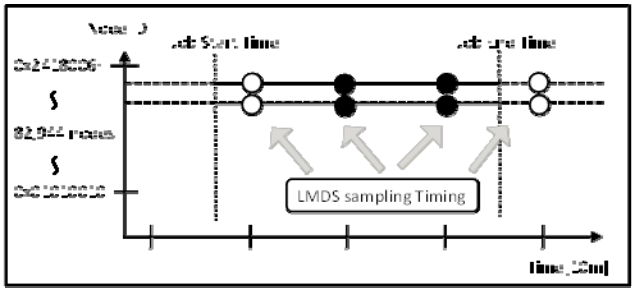

図 5　ジョブごとの LMDS サンプリング方法

Figure 5 Procedure of LMDS sampling for job

4　解析事例

本章では，第 3 章で構築したデータベースをもとに，実際のジョブのデータを用いて LMDS 負荷状況の解析を行ったので，その結果を報告する．解析を行った期間は，本システムの運用を開始した 2016 年 1 月 15 日から 2017 年 2 月 13 日中に計算を開始したジョブが対象である．

4.1 追い出し事例

LMDS 負荷が高いと如何なる現象が起こり得るのか調べた．図 6 は LMDS 負荷が高いジョブとしてピックアップした約 2,000 並列規模のジョブについて，LMDS 負荷の時間変化と，その他 2 本のジョブの LMDS 負荷の時間変化をプロットしたものである．実線のピークは，LMDS 負荷が最初の 10 分で約 35,000[OPS]に達し，その後も，システム全体の LMDS 負荷（点線）を占有して，約 25,000[OPS]で一定の処理速度を維持して推移していることが分かる．またこのジョブの命令実行数の内訳を調べると，10 分単位の全てのノードのサンプリング点にて同じ回数の open/close /getattr の処理をしていることが分かった．これは，短い時間に多くのファイル open/close を繰り返し，システムの限界性能となる 40,000[OPS]を占有し，処理しきれない命令が，遅延して一定のスピードで処理されているものと解釈できる．

この解析から現在行われている 10 分間隔のサンプリングにて，LMDS 負荷が飽和状態となる性能は，約 25,000[OPS]であるといえる．また破線，一点線は別のジョブで，少し小さいノード規模の計算であったが，実行開始後しばらくは LMDS の命令処理が行われていない．しかし，実線のジョブが終了したと同時に処理が開始しており，最終的にはジョブの実行時間制限により，処理を完了することなく終了している．これにより，先行する LMDS 負荷の高いジョブは，他のジョブの遅延を引き起こす可能性があることがわかる．また並行して行わるステージング処理についても LFS や GFS 上の open/close 命令処理が発生するため遅延などの影響があると予想される．

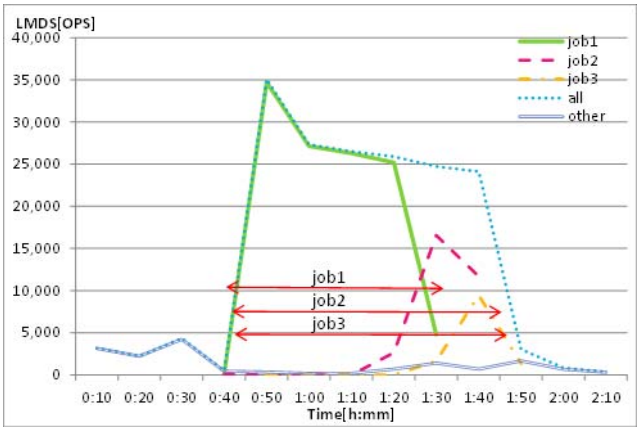

図 6　ジョブによる LMDS 占有の影響

Figure 6 Impact of occupation of LMDS by job

4.2 小規模ジョブの解析

4.1 節で調べた LMDS 負荷の高いジョブは 2,000 並列程度と必ずしも大並列のジョブでなかったことから，小並列で LMDS 負荷の高いジョブについて評価した．

LMDS の負荷は，現状では限界性能と比べて高くないが，LMDS 負荷がウィークにスケーリングしているとし，使用するノード数に比例して増大すると仮定した時に，25,000[OPS]を越える時のノード数を算出した．図 7 は 25,000[OPS]を越え

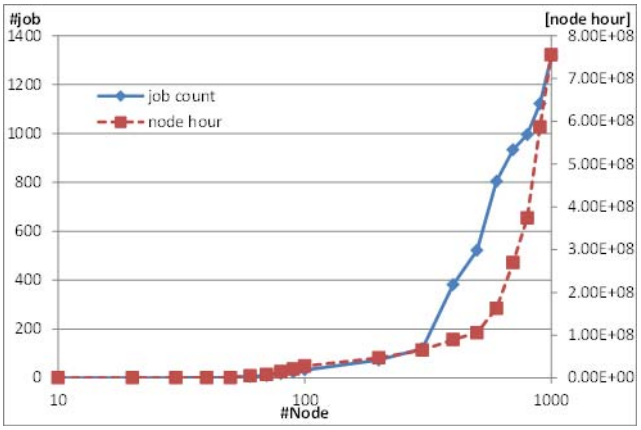

図 7　25,000[OPS]を越えるノード数のジョブの数とシステム使用量

Figure 7 Number of jobs and system usage exceeding 25,000 [OPS]

るノード数のジョブ数とノード時間積である．実線がジョブ数で，破線がノード時間積である．既に 25,000[OPS]を越えているジョブも含まれるが，使用ノード数が 60[node]から LMDS 負荷の高いジョブが現れ始め，3,000[node]を越えると爆発的に増大するのが分かる．

図 8 はジョブの並列数が増えた時に，LMDS の負荷状況がどの程度増大するかを算出しプロットしたものである．図 7 での解析と同様に LMDS アクセス数はノード数に比例すること仮定して算出している．ジョブのノード規模が現状の 2 倍になると LMDS 過負荷のジョブ数は 6 倍以上に増える．更に過負荷状態の時間は，約 14 倍に増えることが想定される．この時間の算出には，過負荷な状態が単一のジョブで生ずるとは限らないことから，同期間のシステム全体の過負荷時間を使用した. LMDS 負荷が高い時間が増大するだけでなく，全体に LMDS 負荷が増えるため，溢れた命令処理に必要な時間が，別途余計にかかるとして算出した効果も含まれる．また過負荷状態のノード時間積については, 28 倍に増大することが予想される．ノード数が 2 倍に増えた分を考慮して算出した資源量である．全体の実行時間は，使用するノード数にストロングにスケーリングして短くなると仮定すると，LMDS の処理時間に相当する効果以外は，トータルの計算資源の増大はない．このことから，現状のアプリケーションのまま I/O 処理を改善することなく，より高並列な計算を行うと，今後，LMDS 過負荷な状態が更に顕在化するものと予想される．

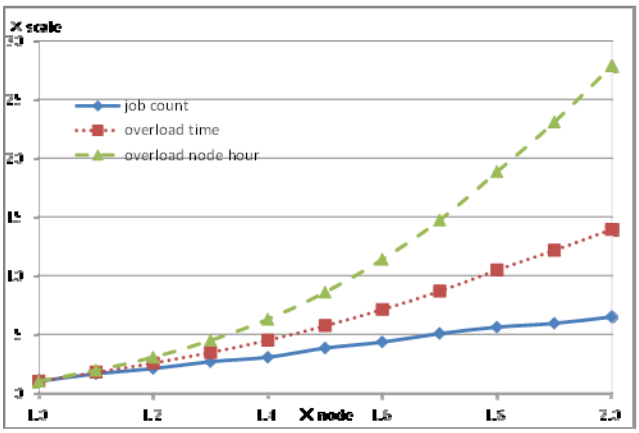

図 8　ジョブノード数が増えた時の過負荷状態の変化

Figure 8 Changes in overload status for the number of job nodes

4.3 ノード時間積への影響

システムの LMDS 負荷が 25,000[OPS]を越えると，その間，自身の計算を含め他のジョブも LMDS アクセスが出来なくなるため，現状の遅延時間並びに影響を受ける資源量を見積もった．資源量の見積もりには，LMDS 負荷が 25,000[OPS]を越えている時間に使用していたノード数を掛け合わせた．表 1 は現状の LMDS 負荷が飽和していた時間並びに資源量の情報である．1 年強の期間で，LMDS 負荷が飽和していたジョブ数は 24 であり，その過負荷時間は 12.8 時間であった．これを使用した計算資源に換算すると，該当するジョブのノードを用いた算出では，約 42,000 ノード時間に相当する．また他のジョブにも影響を与えたとして，全システムに影響があると仮定すると，約 1,000,000 ノード時間に相当すると算出された．

表 1　LMDS 過負荷状態の期間と資源

Table 1 Time and resources of overload of LMDS

LMDS 過負荷状態	数値
ジョブ数	24 [jobs]
ジョブ時間	12.8 [hour]
システム時間	57.7 [hour]
ジョブノードジョブ時間積	41,988 [node hour]
システムノードジョブ時間積	1,064,448 [node hour]

LMDS 負荷が飽和している間は，ファイルの open/close などの処理をしない限り I/O 自体の性能には影響がないため，他の全てのジョブのノード時間積が無駄になるわけではないが，その間遅延が発生している可能性は否定できない．

また LMDS の過負荷な状態は必ずしも 1 つのジョブによって引き起こされるわけではないため，同期間に「京」全体で LMDS 負荷が 25,000[OPS]を越える時間を調べると，57.7 時間と 5 倍近く多く見積もられた．このことは，複数のジョブが要因で LMDS 負荷が高くなる現象が少なくないことを示す. 図 8 の解析結果を見ると, ジョブの LMDS 負荷が 2 倍になると，LMDS 負荷が飽和するジョブが 6 倍になることから，LMDS 高負荷ジョブの予備軍が多く存在することが分かる．これらのジョブが重なり合うと，相乗効果で影響を与える可

能性があることが予想される．但し，データベースに登録されている情報は，ジョブ終了したタイミングで採取されるため，システムの停止を引き起こしたジョブの情報は含まず，その影響も考えられる．

5 まとめと考察

今回，ファイルシステムの遅延から，ジョブによる MDS 負荷情報の解析を行った. この解析を行うために LMDS 負荷情報をジョブ単位で集計し蓄積するシステムを構築した．今回構築したデータベースを用いて解析すると，LMDS 負荷が高い状態になると自分のジョブの実行時間が延びるだけでなく，他のジョブの実行時間へも影響があることがわかった．更に，ノード数が大きくなると将来的に MDS 負荷が高くなると想定されるジョブが他にも多々あることもわかった．

ループバックデバイスを用いることで，メタデータアクセス処理を OSS 内に局在化できるため，MDS 負荷を下げることが可能である．実際には多くのジョブが，既にランク番号ディレクトリを用いたファイル I/O をしている．このため第 4 章で行った，LMDS 負荷がノード数に比例するという仮定での解析は必ずしも実体に即しているとは限らない．しかし今回の解析で，比較的 LMDS の負荷が高いと見積もられたジョブの多くは，何らかの理由でループバックデバイスを利用していない可能性高く，予測精度はそれほど低くはないと思われる．

MDS の性能は，演算処理性能に依存する．このため，その能力を高めるためには，より処理速度の早いサーバに置き換えるなどの対応が可能である．しかし，システムの大幅な変更を伴うため，既存のシステムでは現実的な解決策とは言い難い．

今後これらの解析から，実際に LMDS 負荷が高いジョブに対して，ループバックデバイスへの誘導やファイルアクセスの方法の見直しなどをサポートすることで，システムの有効利用につなげていくことを期待する．但し，共有ファイルシステムを使わざるを得ないアルゴリズムを採用しているアプリケーションも少なからず存在し，更には現状のループバックデバイスでは，一部のジョブ形態並びに MPI-IO や MPI Spawn などの機能に対応していないため，アプリケーションだけでなく，システム開発と協力して取り組むべき課題といえる．

本報では，計算ノードに関する LMDS の負荷情報に着目することで，ジョブによる LMDS の影響についての解析を行った．その他 LMDS に負荷を与える処理として，ステージング処理が考えられる．4.1 節ではジョブがステージング処理に与える影響について議論したが，逆にステージング処理がジョブに影響を与えることも考えられる．この解析には，GIO からの LMDS 負荷を解析することで，評価することが可能であり，今後の課題である．但し，ステージング処理によって，ループバックデバイスにファイル転送した場合，LMDS 負荷は増大せず，共有領域に並列規模に比例する大量のファイル数を転送することは，通常の使用方法では考えられないため，その影響は小さいと考える．

謝辞

本報告に際し，庄司文由氏，宇野篤也氏，山本啓二氏，辻田祐一氏，井上文雄氏をはじめとする理化学研究所計算科学研究機構運用技術部門の諸氏，松井秀司氏をはじめとする富士通株式会社 SE の諸氏に感謝します．本論文の結果は，理化学研究所計算科学研究機構が保有するスーパーコンピュータ「京」によるものです．

参考文献

[1] Sakai, K., Sumimoto, S. and Kurokawa, M.: High-Performance and Highly Reliable File System for the K computer, Fujitsu Sci. Tech. J., Vol. 48, No. 3, pp. 302-309 (2012).

[2] Ajima, Y., Inoue, T., Hiramoto, S., Takagi, Y. and Shimizu, T.: The Tofu Interconnect, IEEE Micro, Vol. 32, No. 1, pp. 21-31 (2012).

[3] 宇野篤也, 庄司文由, 横川三津夫："ファイルステージングのあるジョブスケジューリングの評価", IPSJ SIG Technical Report, Vol. 2012-HPC-136, No. 22 (2012).

[4] 山本啓二, 宇野篤也, 塚本俊之, 菅田勝文, 庄司文由: "スーパーコンピュータ「京」の運用状況", 情報処理, Vol. 55, No. 8, pp. 786-793 (2014).

[5] Sumimoto, S., Matsui, S., Sakai, K., Sueyasu, F., Shoji, F., Uno, A., Yamamoto, K.: "Metadata Access Reduction of Large Scale Lustre Based File System", The Lustre® User Group conference (LUG2015), Denver, USA (2015).

[6] 辻田祐一, 義崎竜彦, 山本啓二, 末安史親, 宮

崎亮二, 宇野篤也: "「京」のファイルシステムの運用改善への取り組み", IPSJ SIG Technical Report, Vol. 2016-HPC-156, No. 12, pp. 1-10 (2016).
[7] 黒田明義, 井上俊介, 小山謙太郎, 関澤龍一, 南一生: "「京」での運用効率改善・省電力に向けたジョブ解析システムの開発", IPSJ SIG Technical Report, Vol. 2016-HPC-156, No. 7, pp. 1-6 (2016).