Vol.2017-HPC-160 No.24
2017/7/27

# 「京」のファイルシステム障害の分析

宇野 篤也[1,a)]　肥田 元[2]　関澤 龍一[3]　辻田 祐一[1]　山本 啓二[1]

**概要**：スーパーコンピュータ「京」は 2012 年 9 月の共用開始から 5 年近くが経過した．「京」のシステムの故障率は低く稼働率は高い状況が続いているが，ハードウェアの故障以外のシステムトラブルに起因する障害は少なからず発生している．障害の原因は様々であるが，最近の障害はファイルシステムに関係するものが大半を占めている．特にローカルファイルシステムに関する障害は，復旧作業においてシステム停止を伴うことが多くその影響は無視できない．今回，「京」で発生したファイルシステム障害について分析を行ったので報告する．

## 1. はじめに

スーパーコンピュータ「京」は，理化学研究所と富士通株式会社が共同開発した汎用並列スーパーコンピュータで，運用は理化学研究所 計算科学研究機構（AICS）が行っている．「京」は 2012 年 9 月から共用を開始し 5 年近くが経過した．システムの故障率は低く稼働率は高い状況が続いている．**図 1** に共用開始後の年度毎の稼働率を示す．ここで示す稼働率は以下の式で算出している．

稼働率 ＝（全時間－予定された保守の時間－障害等による停止時間）／（全時間－予定された保守の時間）

稼働率は低い年でも 96.4%と高い水準にあるが，ハードウェア故障以外の原因による障害は少なからず発生している．共用開始直後はシステムソフト（ジョブマネージャやスケジューラ，資源管理サービス等）に関係する障害が多く発生していたが近年は減少傾向にあり，ファイルシステムに起因する障害が多くを占めるようになってきた．「京」の場合，ファイルシステムに関する障害，特にローカルファイルシステムに関係する障害は復旧するためにシステムを停止させざるを得ないことが多く，安定したシステム運用のためにはファイルシステムに関する障害の発生を減少させることが重要となっている．

今回，「京」で発生したファイルシステムに関する障害について分析を行ったので報告する．

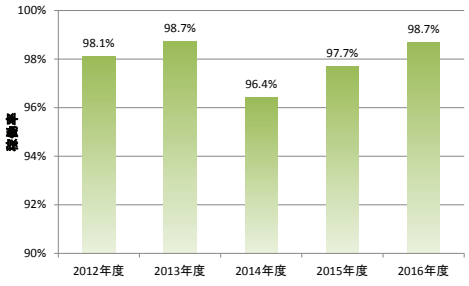


**図 1**　システムの稼働率

## 2. 「京」の概要

### 2.1 「京」のシステム構成

最初に「京」のシステム構成について説明する．**図 2** に「京」のシステム構成の概要を示す [1]．「京」は，82,944 台の計算ノードと 1.27PiB のメモリ（計算ノードあたり 16GiB），5,184 台の I/O ノード，30PB のグローバルファイルシステム (GFS) [*1] と，11PB のローカルファイルシステム (LFS) から構成されている．GFS にはユーザファイルが置かれ，LFS はジョブ実行時の一時領域として使用される．GFS と LFS 間のファイル移動をファイルステージングと呼び，ジョブスクリプトに記載された指示に従って

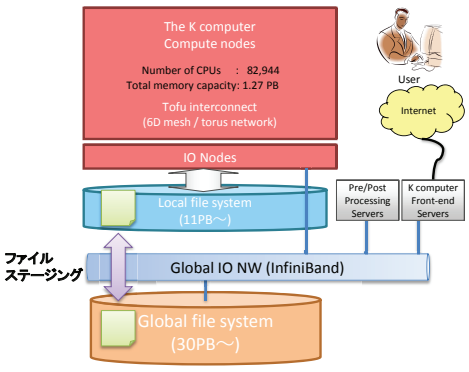


**図 2**　「京」の構成

1　国立研究開発法人理化学研究所 計算科学研究機構
2　株式会社富士通ソーシアルサイエンスラボラトリ
3　富士通株式会社
a)　uno@riken.jp

*1　現在は 40PB

システムが自動で実行する．

GFS と LFS には，富士通が Lustre をベースに機能拡張を行った FEFS (Fujitsu Exabyte File System) が採用されている．GFS は障害発生時の影響を少なくするために複数のボリュームで構成されているが，LFS は全計算ノードからのアクセスを可能とするために 1 ボリュームで構成されている．**表 1** に LFS/GFS の構成を示す [*2]．LFS は帯域重視，GFS は容量重視の構成となっている．

**表 1** ファイルシステム構成

| | GFS | LFS |
|---|---|---|
| OSS 台数 | 90 | 2,592 |
| OST 台数 | 2,830 | 5,184 |
| HDD 容量 | 2TB | 300GB |
| 全体容量 | 30PB〜 | 11PB〜 |
| RAID 構成 | RAID6 | RAID5+0 |

### 2.2 「京」の保守

「京」では保守として以下の 3 種類が定義されている．

- 即時保守
  システム運用が継続できないような重大障害が発生した場合に，可能な限り速やかに実施する保守．
- 定期保守
  比較的小規模かつ軽微な障害のうち，復旧のための作業がシステム運用に大きな影響を与えないものに対して，実施前にあらかじめ期日等を定めた上で実施する保守．予定した保守時間を超過した場合は，超過部分を即時保守として扱う．
- 予定保守
  施設設備の点検等に伴うメンテナンスで，定期的（2 回/年，5 日程度/回）にシステムを全系停止し実施する保守．

定期保守と予定保守は予め予定されている保守であり，即時保守が障害等による停止時間に相当する．即時保守時間を減らすことが，稼働率を上げることになる．

## 3. 「京」での障害発生状況

共用開始以降に「京」で発生した，障害の年度毎の件数 [*3] と障害からの復旧に要した時間（システムが停止した時間）を**図 3** に示す．

システムが停止した時間はシステム全体が停止した場合を基準に計算しており，部分的にシステムが停止した場合は，停止した範囲に応じて計算している．例えば，計算ノードの半分が 1 時間停止した場合は 0.5 時間として計上している．

[*2] GFS は共用開始時点の構成
[*3] システム停止となった障害が対象．コンパイラ障害等は対象外

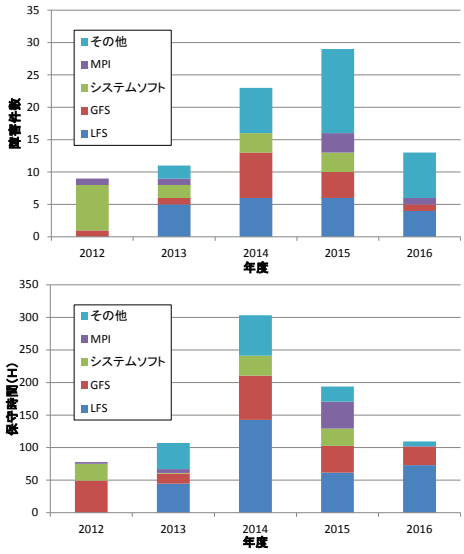


**図 3** 障害発生件数と保守時間

「京」で発生した障害は大別すると

- ファイルシステム（GFS/LFS）
- システムソフト（ジョブマネージャやスケジューラ，資源管理サービス等）
- MPI
- その他

に分類することができる．

発生件数をみると，その他に分類される障害が多くなっているが，この大部分は前述の定期保守が予定時間を超過した場合である．「京」は共用開始から 5 年近くが経過し，システムソフトや MPI 関連の障害は年々減少傾向にあるが，ファイルシステムに関する障害はあまり減少していないことがわかる．これらファイルシステムに関する障害の内容について分析した．

## 4. ファイルシステムに関する障害

「京」の共用開始以降に発生したファイルシステムに関係した障害を調べたところ，以下のような障害（問題）が発生していた．

- Bad Block
- InfiniBand ケーブル障害
- アクセス集中問題
- Loopback 障害
- evict 問題
- ロック競合問題
- GFS のレスポンス低下問題

それぞれについて説明する．

### 4.1 Bad Block

「京」のファイルシステムは RAID で構成されていて，OST のハードディスクが故障した場合でもサービスを停

止することなく復旧することができる．しかし，リビルドの最中にさらにハードディスクが故障するとリビルドに必要なデータを読みだすことができなくなり，リビルドに失敗することになる（GFS は 3 台以上，LFS は 2 台または 3 台以上同時に故障した場合）．この読み出せなくなったブロックを Bad Block と呼ぶ．「京」の場合，リビルドに時間がかかる場合があり（十数時間），Bad Block が発生する確率は低くない．Bad Block が発生した場合，その領域に書き込まれていたデータが通常のファイルであればそのファイルが復旧不可能となるだけですむ．しかし，システムデータ，特にファイルシステムに関する管理データが書き込まれていた場合には，ファイルシステムに不整合が発生する可能性があることになり，そのまま運用を継続することはできなくなる．この場合はファイルシステムの再構築を行う必要があり，システムを停止することになる．

共用開始当初は Bad Block が発生した領域にどのようなファイルが書き込まれていたか判断することができなかったため，管理領域の可能性を考慮し Bad Block 発生時にはその都度ファイルシステムの再構築を実施していた．しかし，頻繁にファイルシステムを再構築することはできないため，Bad Block が発生した領域の利用状況を判断する手法を確立した．これにより，管理領域でかつ使用領域の場合を除き運用を継続することができるようになった．例えば，Bad Block が発生した領域が LFS のデータ領域で使用中なら，その領域を使用していたジョブのみを再実行させて問題の領域を開放させ，システムの運用は継続するといった対応を行う．

### 4.2 InfiniBand ケーブル障害

「京」のファイルステージング処理は，ジョブの実行とは非同期に行われる．この時，スケジューラは転送するファイルサイズからステージング処理に要する時間を見積もり，ステージング処理のスケジューリングを行う．そのため，見積もり時間を大幅に超過すると後続のジョブの実行に大きな影響がでる．見積もり時間を超過する原因はいろいろとあるが，ハードウェア障害に起因するものとして，InfiniBand(QDR) の性能低下が原因となる事象が発生した．

「京」の LFS と GFS は IB で接続されており，ケーブルに問題が発生した場合にはログにメッセージが出力される．実際に障害が発生したかどうかは，ログに出力されたメッセージの内容や出力頻度から判断されるが，故障と判断されない状態にもかかわらず転送性能が低下する状況となっていた．この場合，実際にデータの転送速度を測定しない限り異常を検出できないため，ステージング処理等で問題が発生した場合に原因を見つけるのが困難になる．特にステージング処理の見積もり時間超過は，ファイルシステムのアクセス集中による Read/Write 性能低下などの原因でも発生することがあるため，原因の切り分けが難しい．

### 4.3 アクセス集中問題

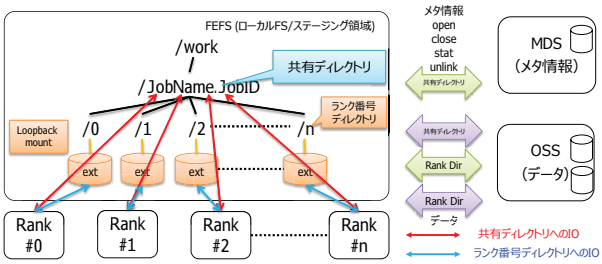


**図 4** ランク番号ディレクトリ

「京」の LFS は単一のファイルシステムで構成されているため，高並列ジョブのファイルアクセスで問題が発生することがある．MDS の過負荷と特定 OST へのアクセス集中である．例えば，10,000 ノードのジョブがランクごとに 100 ファイルを作成した場合，ジョブ全体では 100 万ファイルが作成され MDS への負荷が非常に高くなり，単純なファイル作成/削除でも非常に時間がかかる場合がある．また，各ランクから単一のファイルに同時にアクセスを行うと，MDS と OST の両方にアクセスが集中し処理に長い時間がかかる．例えば，全計算ノード（82,944 台）を使用して MPI プログラムを実行するような場合である．

これらの問題を解決するために，ランク番号ディレクトリを導入した [2]．**図 4** にランク番号ディレクトリの概要を示す．ランク番号ディレクトリでは，ランク毎に Loopback ファイルシステムが作成され，各ランクのファイルアクセスを分散することが可能となる．**図 5**，**表 2** に Loopback 適用前後の MDS 負荷を示す．これらからわかるように，Loopback ファイルシステムを導入することにより，MDS 負荷を下げることができている．

### 4.4 Loopback 障害

前述の通り，アクセス集中問題を解決するために Loopback ファイルシステムを導入したが，並列度の高いジョブの実行時に Loopback の前処理が予定時間内に終了しないという障害が発生した．Loopback の前処理の各処理の所要時間を分析したところ，Loopback の作成に必要な OST

**表 2** Loopback 適用前後の MDS 負荷

| | 適用前 –13/12/27 | 適用後 1 –14/6/29 | 適用後 2 –15/2/28 | 適用後 全期間 |
|---|---|---|---|---|
| 平均 MDS 負荷 (%) | 25.1 | 8.21 | 6.36 | 7.13 |
| 50%超の発生回数/日 | 2.32 | 0.12 | 0.02 | 0.06 |
| 70%超の発生回数/日 | 0.68 | 0.04 | 0.00 | 0.02 |

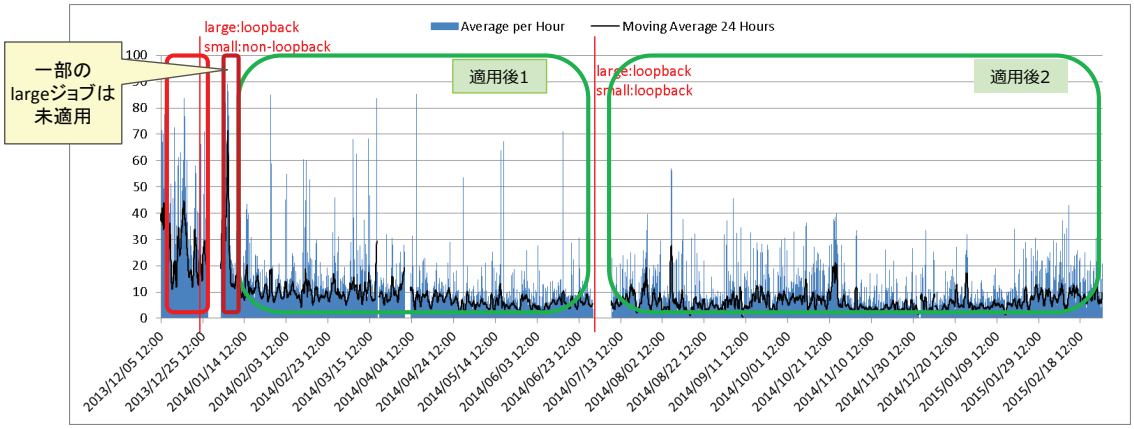


**図 5** Loopback 適用前後の MDS 負荷

情報の取得と Loopback の作成処理に時間がかかっていることが原因であった．OST 情報の取得処理は，他の処理とまとめるなどの最適化をおこない処理時間の短縮を行った．Loopback の作成処理は，イメージファイルへのアクセス性能低下が問題であった．これは，loopback のイメージファイルをジョブ毎に作成されるディレクトリの直下にまとめて配置していたため，同一ディレクトリに対するアクセス集中が原因で発生していた．これを回避するために，ランク毎のディレクトリを作成しそこにイメージファイルを配置するとともに，処理内容の最適化を行った．これら最適化を行った結果，Loopback の前処理時間は半分近くに短縮された．

### 4.5 evict 問題

evict とは一定期間ファイルサーバから反応がなかった場合など，正常に処理ができない場合にクライアントを切り離す処理で FEFS にも実装されている．evict 自体は正常な処理だが，これが発生するとクライアント側では IO 処理がストールするなどの状態になりジョブが異常終了するなどの影響がでるため，なるべく発生しないようにする必要がある．「京」の場合は，クライアントからのリクエストがタイムアウトした場合以外にも，OSS のフェイルオーバーやシステムボード (SB) をメンテナンスで取り外す時にも発生するため，システムを安定稼働させるために解決すべき重要な問題の一つであった．これらの問題を解決するために，SB のメンテナンス手順の見直しとタイムアウトに対する FEFS の改良を実施した [3].

SB を交換する場合，構造上は SB 単位で活性交換することが可能である．交換のため SB を切り離すと Tofu ネットワークが再構成されるが，このとき Tofu の z 軸を共有する同一キャビネット内の他の SB で実行されているジョブで evict が発生し，ジョブが異常終了することがある．これを防ぐため，SB を交換する際には SB 単位ではなくキャビネット単位で運用を停止する運用とした．これにより SB 交換時の evict 発生を防ぐことができるが，停止ノード数が SB 単位の 4 ノードからキャビネット単位の 192 ノードとなるというデメリットが発生する．**図 6** に SB 交換手順の変更前後の evict 発生頻度を示す．手順変更により発生頻度が減っていることがわかる．

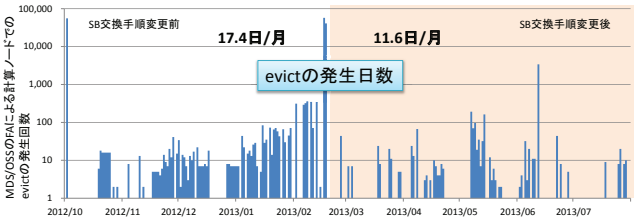


**図 6** SB 交換手順の変更前後の evict 発生頻度

また，クライアントからのリクエストのタイムアウト発生に対する FEFS の改良として，クライアント側のリクエスト再送タイミングの改善と，サーバ側で高負荷時でもリクエストを処理できるように受信専用スレッドの確保を行った．これらの改良により，evict の発生頻度は改良前の 1/72 まで減らすことができた．

### 4.6 ロック競合問題

一つのファイルに対して多数のプロセスから同時に書き込みを行う場合，一般的に排他制御を行うためにファイルロックが行われる．並列度が低い場合は大きな問題とならないが，例えば「京」の全ノードから同時に 1 ファイルに対して書き込みが発生した場合，1 プロセスで 10ms 要したとしても全ての処理が終わるまでに約 830 秒かかることになる．この間，SIGKILL 等のシグナルも一切受け付けない状態になるため，予定時間を超過してもジョブを強制終了させることができないなど，運用に非常に大きな影響がでる．これは「京」に限らず大規模なシステムで発生しうる問題で，システム側での対応が困難でジョブ側で競合が発生しないようにするなどの対処が必要となる．

### 4.7 GFS のレスポンス低下問題

ログインノードにおいて一時的に GFS へのアクセスが遅延するという問題が発生した．この問題が発生している間は，単純なファイル編集のような操作でも長時間待たされるといった影響が出る．**図 7** に GFS のアクセスが遅延が発生した事例を示す．

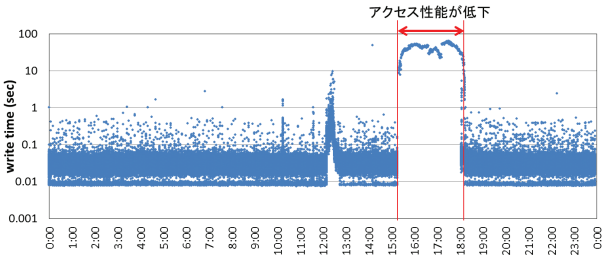


**図 7** GFS のアクセス遅延の事例

これは GFS で 1MiB のファイルの書き込みに要した時間を示していて，アクセス性能が急激に低下していることがわかる．アクセス性能が低下していた時間帯を調査した結果，ジョブのファイルステージングが行われている際に発生していることが判明した．小さなファイルではこの問題は発生せず，MDS に余裕があるにもかかわらず規模の大きなファイルがステージングされている際に顕著化していることから，GFS の OSS/OST が高負荷状態になりレスポンスが低下したものと推測した．そこで，ステージアウト時のストライプカウント設定と FEFS が有する QoS 機能によるファイルアクセス時の負荷バランス制御を行うことで運用改善を図った [4],[5].

ストライプカウントではファイルを書き込む際に使用する OST の数を指定する．1 が指定された場合は特定の OST に集中して書き込みが行われるが，2 以上が指定されると指定された個数の OST でストライピングアクセスが行われ負荷が分散される．共用開始直後はユーザが指定しない限りストライプカウント 1 で書き込まれており，これがレスポンス低下の一因となっていたと推測された．そこで，ステージアウト時にシステム側で自動的にストライプカウントを指定するように設定を行った [*4]．**図 8** に負荷バランス設定なしでストライプカウント 1($C_s$=1) の場合とストライプカウント 32($C_s$=32) の場合のステージング処理時の GFS レスポンス時間の分布を示す．ストライプカウントを 32 に設定することで IO 処理の負荷バランスが改善されていることがわかる．なお，現在のストライプカウントの設定は運用上の経験から設定したもので，現在適切なストライプカウントの設定方法について引き続き検討を続けている [5].

また，FEFS の IO 処理は，基本的に FCFS に基づいて行われるため，負荷の高いジョブが IO 処理（サービススレッド）を専有するという問題が起きうる．特にサイズの大きなファイルを特定の OST に書き込むような処理が複数あると，それらの処理が完了するまで他の処理が待たされレスポンス低下の一因となる．この問題に対し，FEFS の QoS 機能を利用して対応を行った．この機能により，IO リクエスト元毎にサービススレッドを割り当てたり，1 ユーザに発行可能な IO リクエストの上限管理を行うことが可能となる．**図 9** に負荷バランス（フロントエンドサーバとそれ以外に対するアクセス比）を設定しない場合と 20:80 に設定した場合の GFS レスポンス時間の分布を示す．負荷バランスを設定することにより，レスポンスが改善されていることがわかる．

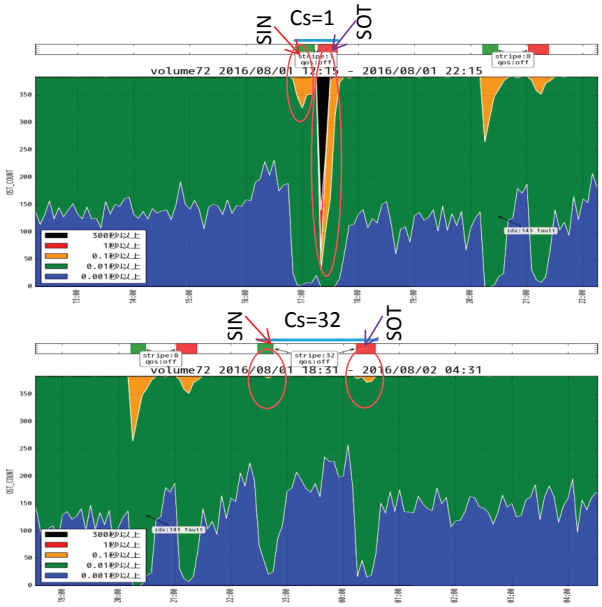


**図 8** 負荷バランス設定無しでストライプカウントを変えた場合の GFS レスポンスの時間の分布 (12GiB/ファイル，48 台の GIO を使用)

## 5. 障害発生の予防策

システム運用に影響がでるような障害が発生した場合，如何に早くシステムを復旧させることができるかが重要となるが，障害自体を発生させないことができれば，その方が望ましい．障害の発生を防ぐには，障害発生の兆候を事前に検出することが必要となる．特に「京」ではファイルシステムに関連する障害は復旧のためにシステムを停止させる必要がある場合が多くあり，障害発生を未然に防ぐことは重要である．

ハードウェア故障の場合は，様々な情報が出力されているログを監視することで検出することができる．例えば，IB ケーブルの場合なら通信時に発生するエラーの種類と発生回数を，ハードディスクならリカバリー可能なリードエラーの回数などを監視することである程度の障害の発生を予測することが可能である．実際，これらは「京」でも

[*4] ファイルサイズを 16MiB で割った値か 32 の小さい方を指定

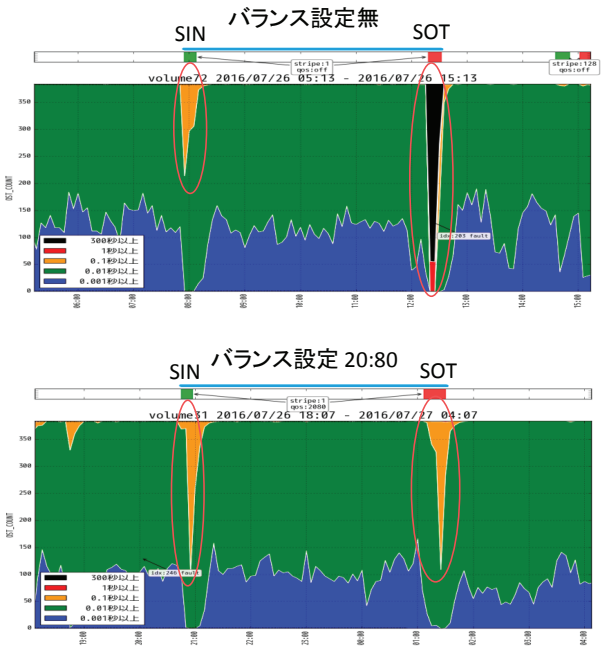


**図 9** 負荷バランスを設定しない場合と 20:80 に設定した場合の GFS レスポンスの時間の分布 (12GiB/ファイル，$C_s$=1，96 台の GIO を使用)

実施しているが，故障の兆候を正確に把握することは難しく，故障発生の判断条件について検討を行っている．

ソフトウェア障害の場合，システムソフト等のバグによる障害は未然に防ぐことは困難であるが，ジョブにより MDS が高負荷状態になるような障害については，ハードウェア故障と同様にログ等の監視で検出することが可能と考える．一例として，過去に LFS がアクセス不可状態になった際の計算ノードからのリクエスト数と計算ノードの FEFS のクライアントのタイムアウトの発生状況を**図 10** に示す．この図からわかるように，クライアントリクエストのタイムアウトが多発したあと LFS がダウンしており，タイムアウトが発生し始めた段階で高負荷状態を引き起こしているジョブを停止することができれば LFS のダウンを未然に防ぐことが可能となる．この例のようにログ等を監視することで，障害の発生を検出することが可能と考えているが，そのためには，リアルタイムに各種ログを監視・分析して問題の発生を検出する仕組みが必要となる．現在，監視体制の構築と検出条件の検討を行っている．

## 6. おわりに

本稿では，「京」で発生したファイルシステムに関する障害について述べた．「京」は 2012 年 9 月の共用開始から 5 年近く経過したが，システムの故障率は低く稼働率は高い状況が続いている．共用開始直後はシステムソフトに関係する障害が多く発生していたが，近年はファイルシステムに起因する障害が多くなっている．特にローカルファイルシステムの障害はシステムダウンに繋がることが多く，安

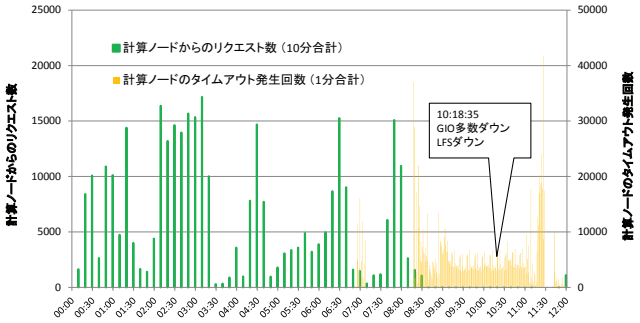


**図 10** LFS ダウン時の状況

定したシステム運用のためには障害の発生を防ぐことが重要である．

ハードウェア故障については，ログに出力されるメッセージを精査することでその発生を予測することができると考えているが，判断条件を現在検討中である．

ソフトウェア障害については，ソフトウェアのバグによる障害の検出は困難であるが，ジョブに起因するような障害は予測可能と考えている．具体的には，ジョブの実行状態をログ等で監視し，問題が発生しそうな場合に当該ジョブを停止させる．そのためには，リアルタイムに大量のログを監視・分析する必要があり，現在手法を検討している．

これら，障害発生を予測する手法を確立することは様々なシステムで役に立つと考えている．

## 参考文献

[1] 特集：スーパーコンピュータ「京」, 情報処理, Vol.53, No.8, pp.752–807, 2012
[2] Shinji Sumimoto, Shuji Matsui, Kenichiro Sakai, Fumichika Sueyasu, Fumiyoshi Shoji, Atsuya Uno, Keiji Yamamoto, “ Metadata Access Reduction of Large Scale Lustre Based File System ”, LUG2015, 2015
[3] Keiji Yamamoto, Fumiyoshi Shoji, Atsuya Uno, Shuji Matsui, Kenichiro Sakai, Fumichika Sueyasu, Shinji Sumimoto, “ Analysis and Elimination of Client Evictions on a Large Scale Lustre Based File System ”, LUG2015, 2015
[4] 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，情報処理学会研究会報告 Vol.2016-HPC-156 No.12, 2016
[5] Y. Tsujita, T. Yoshizaki, K. Yamamoto, F. Sueyasu, R. Miyazaki, and A. Uno, “Alleviating I/O Interference Through Workload-Aware Striping and Load-Balancing on Parallel File Systems,” LNCS 10266, pp. 315–333, 2017