Vol.2015-HPC-152 No.3
2015/12/16

ノード故障によるジョブスケジューリングへの影響評価

関澤 龍一[1,a)]　宇野 篤也[2]　山本 啓二[2]　若林 大輔[3]　肥田 元[3]　池田 直樹[3]

概要：
スーパーコンピュータや PC クラスタといった HPC システムでは，高並列化に伴う構成部品数の増加によりシステムの故障率が増加傾向にある．通常，計算ノードの故障がそのままシステムの運用停止につながるようなことはほとんど起きないが，故障ノード数が増加すると運用への影響は無視できないものとなる．故障ノードが発生する毎に保守を行うことで運用への影響を最小限にすることができるが，頻繁な保守作業は運用コストの面から困難である．本稿では，故障ノードの発生が運用に及ぼす影響を最小限にできる保守タイミングについて検討を行った．今回，故障ノードの発生が運用に及ぼす影響としてジョブスケジューリングに着目し評価を行ったので，その結果について報告する．

1. はじめに

スーパーコンピュータや PC クラスタといった HPC システムは，用途の拡大や計算速度性能の向上に伴い，高並列化する傾向にある．例として，TOP500 [1] の上位にランキングした HPC システムのコア数を挙げる．2010 年 6 月の TOP500 では，1 位から 10 位までにランクインした HPC システムの平均コア数は約 143,000 であったが，5 年後の 2015 年 6 月の TOP500 では約 837,000 であった．5 年間で 6 倍近くにコア数が伸びており，HPC システムが高並列化の傾向にあることが分かる．高並列化はシステムを構成する部品数の増加につながり，故障数の増加にもつながる．例えば，理化学研究所 計算科学研究機構 (AICS) が運用を行っている「京」では，計算に使用するコア数は 663,552，ノード数は 82,944 と高並列なシステムで [2]，システム全体では 100 万個以上もの部品から構成されている．

通常は計算ノードの故障発生がそのままシステムの運用停止につながるようなことはほとんどないが，故障ノード数が増加するにつれ，運用への影響は無視できなくなってくる．運用への影響を最小限にするには，故障ノードが発生する毎に保守を行えばよい．しかし，頻繁な保守作業を実施することは運用コストの面から難しいため，故障ノードの発生が運用に及ぼす影響を把握し，運用への影響が大きくなる前に保守を行なえることが望ましい．

今回，故障ノードの発生が運用に及ぼす影響についてジョブスケジューリングに着目し評価を行った．着目した理由は，故障ノードの発生がジョブスケジューリングのノード割り当てに大きく影響をするためである．本稿ではその評価結果について報告する．

2. ノード故障とスケジューリング空間

2.1 ノード故障発生時の動作

高並列システムでは，構成の柔軟性とスケーラビリティに優れているという観点から，ネットワークトポロジーに直接網が採用されることが多い．直接網においてノード間通信性能を確保するには，接続しあったノードをジョブに割り当てる必要がある．その場合ノードが故障すると，システム全体の空間内にジョブの使用できない箇所が生じることになるため，ジョブにノードを割り当てる際には，その箇所を避けつつ連続した空間にジョブが埋まっていく．故障数が増えれば増えるほど避けるべき箇所が空間内に点在していき，空間全体の中でジョブが確保できる空間 (以下，スケジューリング空間) が狭まっていく．

先述したノード故障発生とスケジューリング空間について説明する．簡略化のため，10 × 7 の 2 次元でノードがメッシュ接続されたシステム (**図 1**) を例に説明する．故障は (2，1)(6，4) に発生しており，赤色矩形はスケジューリング空間 (2 次元の場合は矩形) を示す．この座標を使用せずにノードを確保する必要があるので，確保できる矩形の形状に制限が生じる．さらに故障数が増加すると，確保できる形状がより狭まる可能性がある．図 1 における最大ス

[1] 富士通株式会社
Fujitsu Limited
[2] 理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
[3] 株式会社富士通ソーシアルサイエンスラボラトリ
Fujitsu Social Science Laboratory Limited
[a)] r_sekizawa@jp.fujitsu.com

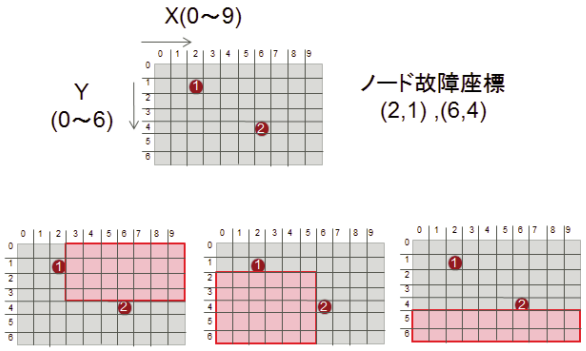

図 1　スケジューリング空間

ケジューリング空間は 6 × 5 の空間となる．

2.2　故障座標と最大スケジューリング空間の関係

スケジューリング空間が変わると，実行されるジョブの順序も変化する可能性が高い．特に，ノード故障が散らばって発生することで最大スケジューリング空間が狭くなると，ノード数の大きいジョブに対する割当の自由度が減ることで実行を開始するまでの待ち時間が長くなり，ジョブスケジューリングに影響が及ぶと想定できる．

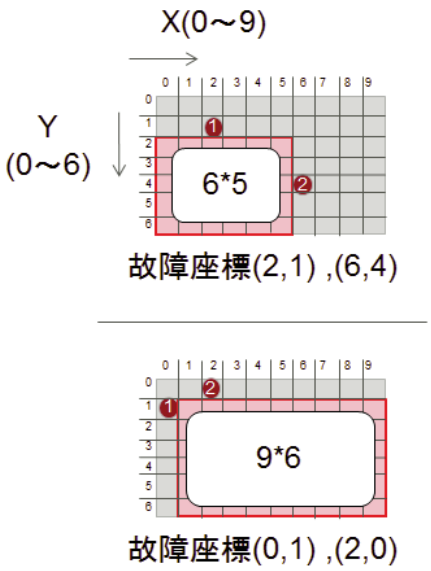

図 2　故障座標と最大スケジューリング空間の関係

図 2 にて，故障座標が異なる場合の最大スケジューリング空間の差について説明する．図 1 と同じ 10 × 7 の 2 次元システムにおいて，一方は (2，1)(6，4) に，もう一方は (0，1)(2，0) にノード故障が発生したと仮定する．前者の場合は最大スケジューリング空間が 6 × 5 であるのに対し後者は 9 × 6 で 2 倍近くの差となる．故障したノードが少ないか位置に偏りがあると，最大スケジューリング空間は広く，故障したノードが広く散らばると，最大スケジューリング空間は狭くなる．故障発生位置によって最大スケジューリング空間に開きができることがわかる．

以上から，ノード故障によって発生するスケジューリング空間の差により，ジョブスケジューリングへの影響がどう異なるかを評価することとした．

3.　評価

評価環境として「京」ジョブスケジューラと同等の機能を持つジョブスケジューラシミュレータを使い，異なるばらつきでノードを故障させた際のスケジューリング状況の変化を評価した．評価には「京」で実行されたジョブの特性を反映させたジョブミックスを使用した [3]．

3.1　ジョブミックス

ジョブミックスは様々なジョブの集合体で，個々のジョブ毎に，投入時刻 (投入間隔)，ノード数，経過時間指定，実実行時間といったパラメータをもつ．ジョブミックス毎のばらつきを考慮し，ランダムに作成した複数のジョブミックスを使って評価した．今回は，「京」で 2012 年 10 月から 2015 年 3 月までの間に実行されたジョブの確率分布マップをもとに生成したジョブミックスを使用した [3]．

図 3 は，「京」オリジナルデータの統計情報を示したものであり，ジョブミックスもこの特性を持っている．

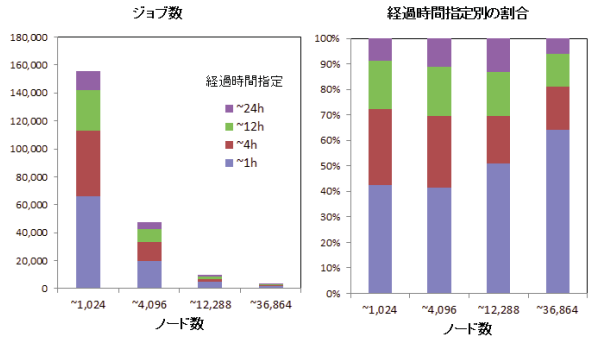

図 3　「京」オリジナルデータの統計情報

3.2　ジョブスケジューラシミュレータ

評価に用いるジョブスケジューラシミュレータは「京」で使用しているジョブスケジューラと同等の機能を持つものを使用した．このシミュレータでは，位置を指定してのノード故障と復旧も可能となっている．本シミュレータでは，新規にジョブが投入されたときと実行中のジョブが終了したとき，ノードの故障・復旧が行われたときにジョブスケジューリングが行われる．また，シミュレータはジョブ毎に統計情報を出力する．統計情報には，ジョブが実行待ち状態から実行状態へ遷移した時刻や終了した時刻，

ジョブが確保したノード数などが記載されている．ジョブ毎の統計情報から，ノード利用率と投入から実行開始までの待ち時間をそれぞれ計算する．スケジューリングアルゴリズムは「京」と同じ FCFS とバックフィルを使用し，連続した直方体のノードをジョブに割り当てている．その他のシミュレーションパラメータは「京」に準拠した．

3.3 ノード故障・復旧スケジュール

シミュレーション期間は 28 日間とし，密度 100%のジョブミックスを 10 セット用意した．ジョブミックスの密度とは，システムの全計算資源に対する投入ジョブのノード時間積の総和の割合を表している．

ノード故障発生と復旧のスケジュールは，開始 24 時間後から 1 ノードずつ 24 時間間隔で発生，6 ノード目が故障した 24 時間後に溜まった 6 ノード全てを復旧させることとした．ここまで丸 7 日間を要する．このセットを 4 回続け，シミュレーション期間である 28 日間にした (**図 4**)．

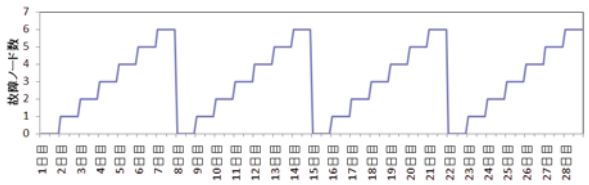

図 4 ノード故障・復旧スケジュール

ノード故障・復旧スケジュールは，最大スケジューリング空間の下限値が，システム全体の 40%,60%,80%をそれぞれ下回らないような，3 種類のスケジュールを用意した．最大スケジューリング空間の下限値とは，6 ノードの故障が溜まった時点での最大スケジューリング空間を表す．**図 5** は，3 種類のスケジュールにおける最大スケジューリング空間の下限値推移の一例を示す．故障は 1 ノードずつ時間間隔をあけて発生させるため，故障が溜まっていく過程で最大スケジューリング空間は徐々に狭くなっていく．6 ノード溜ったときが最も小さいサイズとなる．

最大スケジューリング空間が小さいほど故障位置にばらつきがある．比較する 3 種類の中では，下限値 40%が最も散らばって故障しているパターンである．なお，ジョブミックス毎に異なるノード故障・復旧スケジュールを使用しているため，必ずしも図 5 のように縮小していくわけではない．実際に故障が発生する状況を考慮し，スケジュールにおける故障発生位置はあくまでランダムで，規則的に発生させてはいない．

また，ジョブミックスにおけるジョブのノード数の範囲は，385〜36,864 ノードである．「京」は 82,944 ノードで構成されており，その空間の 40%となると 33,177 ノードに相当するため，それを超えるジョブが投入された場合は，復旧するまで実行できない状況が起こりうる．以上からも，40%の場合が最もジョブスケジューリングによる影響を受けやすいと推測できる．

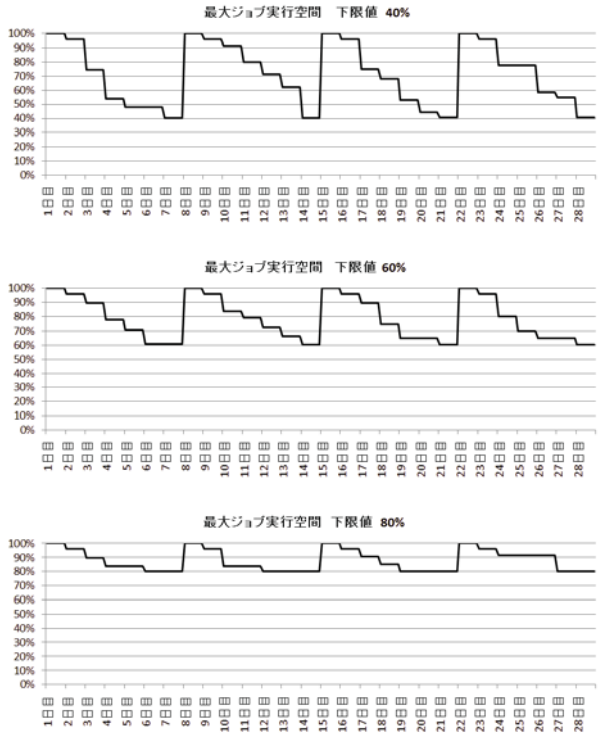

図 5 下限値別の最大スケジューリング空間推移 (一例)

さらに比較用として全く故障させないパターンを加えた，計 4 種毎にそれぞれ 10 セットのシミュレーションを行い，平均値を評価した．

4. シミュレーション結果

ノード利用率とジョブ実行待ち時間をもとに，評価を述べる．ノード利用率やジョブ実行待ち時間の統計範囲は 2 日目から 28 日目までとしている．開始直後はジョブ数が少ないので，定常的になる 2 日目以降を評価対象とした．

4.1 ノード利用率

図 6 に，2 日目から 28 日目までのノード利用率を示す．故障により途中で強制的に戻された際の利用分については，ノード利用率の統計から省いている．

故障を発生させたパターンの中では下限値 80%の利用率は 89.00%で最も高く，次に 60%での利用率が 88.64%，そして 40%の利用率は 88.42%と最も低い結果となった．故障がばらついた場合 (最大スケジューリング空間が狭まった場合) に，利用率が落ち込んでいる．差を比較すると，80%から 60%では 0.36%低下，60%から 40%では 0.22%低下となっており，最大スケジューリング空間が狭まるにつれ利用率も下がっている．故障が発生しないパターンの利用率は 90.84%で，故障発生する場合と比較すると 2%前後の差となった．

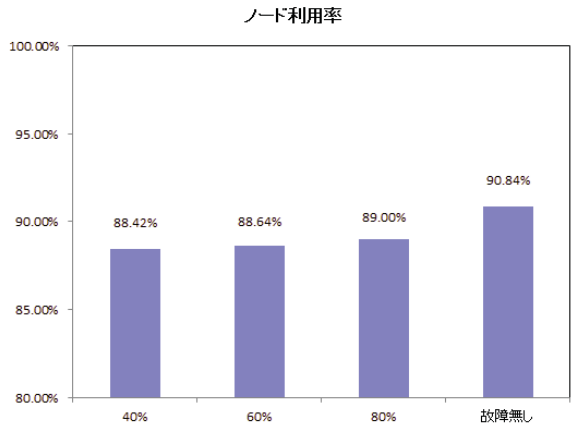

図 6　最大スケジューリング空間別のノード利用率

4.2　ジョブ実行待ち時間とジョブ実行数

図 7 に，ジョブのノード規模と経過時間指定別のジョブ実行待ち時間と実行されたジョブ数を示す．この図は，各ジョブミックスに 2 日目から 28 日目までの間に実行を開始したジョブの，ジョブ実行待ち時間とジョブ数の平均値である．全体から見える傾向は，ノード数の大きいあるいは経過時間指定の長いジョブは最大スケジューリング空間の下限値低下に連動して，ジョブ実行待ち時間が少し伸びていることである．もともと実行されにくいジョブであるノード数規模や時間規模の大きいジョブが影響を受けた結果となったが，それほど大きい差は無い．最大でも，12,289 から 36,864 ノード以下で 24 時間までの経過時間指定ジョブが 10.4 時間の待ち時間差であったが，待ち時間が 100 時間程であることを考えるとジョブ実行待ち時間への影響は少ないといえる．

ジョブ実行数に関しては，最大スケジューリング空間の差による変化はほぼない．故障発生位置の違いにより少々ジョブ実行待ち時間が伸びることがあっても，期間内に実行できるジョブ数には影響を与えない．

4.3　評価

評価前の想定どおり，故障位置が広く散らばると利用率が低下し規模が大きいジョブの待ち時間は伸びたが，最大スケジューリング空間の差によるジョブスケジューリングへの影響差は大きくは無かった．以上から，「京」環境下では 40%から 80%のジョブ実行保障空間の差があってもノード利用率やジョブ実行待ち時間といったジョブスケジューリングへの影響は軽微であるといえる．

理由として，ジョブミックスにノード数の小さいジョブが多かったため，最大スケジューリング空間の大小による影響を受けなかったと考えられる．ノード故障によりスケジューリング空間は分割するが，82,944 ノードである「京」において故障数 6 ノードという少なさだと，スケジューリング空間の分割が粗い．最大スケジューリング空間が小さかったとしても，分割が粗く個々の空間が大きいと，ノード数の小さいジョブにとって影響はない．つまり，ジョブミックスの構成に依存するといえる．

故障数が少なかったことで，スケジューリング空間の分割が粗くなり，ノード数の小さいジョブは影響を受けなかった．最大スケジューリング空間がある値に落ち込むまで故障を発生させるという条件をとることで，ジョブスケジューリングへの影響がより詳細に評価できると考えられる．ジョブミックスやシステムノード数など，今回は「京」を意識した評価環境としたが，スケジューリング空間による影響を評価しやすくするために，簡易的な環境から段階的に評価することも重要である．

5.　関連研究

過去には，3 次元トーラスネットワーク構成である BlueGene/L を使用してノード故障の影響を小さくすることができるジョブスケジューリングアルゴリズムを複数検討し，それぞれに対するジョブスケジューリング性能の評価が行われている [4]．後継の BlueGene/P でも継続して，ノード故障発生状況下において利用率向上やジョブ実行待ち時間の短時間化に与するスケジューリングアルゴリズムが検討，評価された [5]．

これらの研究は，ノード故障に適するスケジューリングアルゴリズムを提案し評価したという試みである．一方本研究では，ノード故障のばらつきに着目し，ジョブスケジューリング性能との関係について評価している．効率的に計算資源を活用するという目的は同じだが着眼点が異なる．

6.　まとめ

本稿では，運用に影響を与えずに保守作業の頻度を抑える手段として，故障したノードの交換タイミングについて着目し，ノード故障が与えるジョブスケジューリングへの影響について評価した．ノード故障発生位置のばらつきにより，スケジューリング空間に差ができたときに，ジョブスケジューリングが影響を受けると推測した．評価環境として「京」のジョブスケジューラシミュレータを使用し，投入するジョブの傾向も「京」に準拠した．

故障により最も大きなスケジューリング空間がそれぞれ 40%,60%,80%となるよう故障・復旧スケジュールを策定して，シミュレーションを行った結果，最大スケジューリング空間が狭くなるほど，ノード利用率が落ち込み，ノード数が大きいあるいは経過時間指定が長いジョブの待ち時間がのびた．しかし，利用率・待ち時間のどちらにおいても 40%と 80%を比較した際の増加幅はあまり大きくはならなかった．今回の評価では最大スケジューリング空間が 40%になるまで故障したノードをそのままにしてもジョブスケジューリングへの影響がないことがわかった．

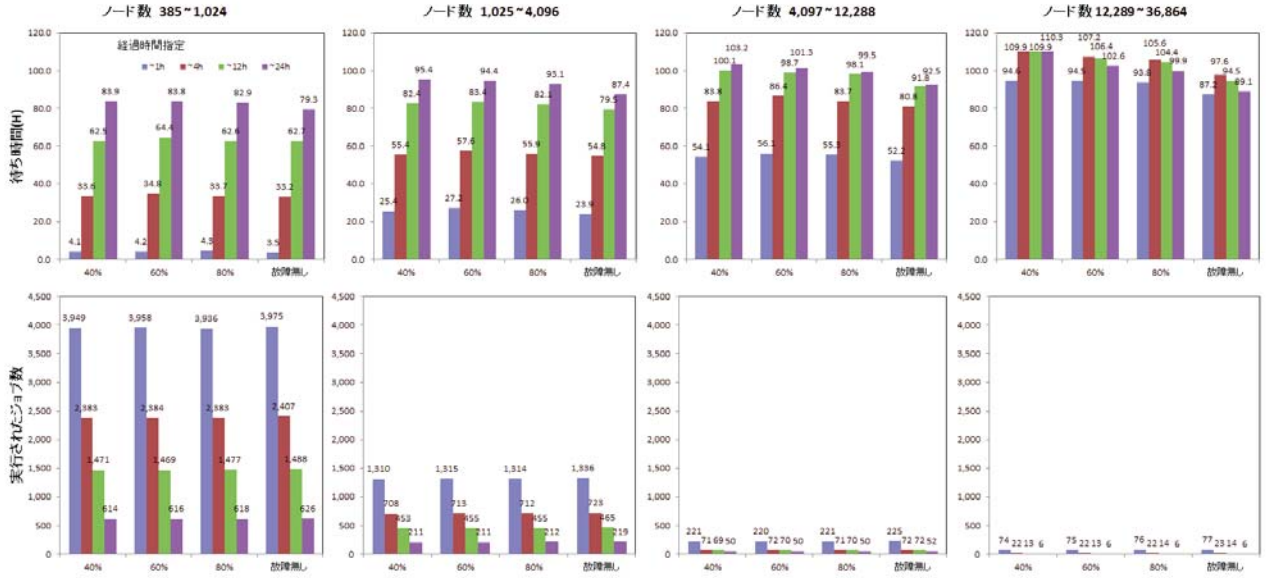

図 7　ジョブ実行待ち時間と実行されたジョブ数

しかし，今回使用したジョブミックスにはノード数の小さいジョブが多く，故障数が少なかったことで分断された個々のスケジューリング空間の大きさに余裕があったことから，ジョブスケジューリングへの影響が小さくなったとも考えられる．最大空間だけではなく個々の空間の大小も加味し，評価することが必要であると考えられる．

今後は，最大スケジューリング空間に加えて，個々の空間の大小やノード故障数にも着目し評価を継続していきたいと考えている．

参考文献

[1] TOP500: TOP500 Supercomputer Sites, Top500.org (online), available from (http://www.top500.org) (accessd 2015-5-20).

[2] Yamamoto,K., Uno,A., Murai,H., Tsukamoto,T., Shoji,F., Matsui,S., Sekizawa,R., Sueyasu,F., Uchiyama,H., Okamoto,M., Ohgushi,N., Takashina,K., Wakabayashi,D. ,Taguchi,Y., Yokokawa,M.: The K computer Operations: Experiences and Statistics., International Conference on Computational Science ICCS2014, pp. 576-585 (2014).

[3] 宇野篤也，関澤龍一，山本啓二，若林大輔，庄司文由：「京」上のジョブの分析とジョブミックス生成手法の提案，情報処理，Summer United Workshops on Parallel, Distributed and Cooperative Processing ，SWoPP2015(2015).

[4] A. J. Oliner, R. K. Sahoo, J. E. Moreira, M. Gupta, and A. Sivasubramaniam. : Fault-aware Job Scheduling for BlueGene/L Systems, Proc. 18th Int'l Parallel and Distributed Processing Symp. (IPDPS04), p.64(2004)

[5] W. Tang, Z. Lan, N. Desai, and D. Buettner : Fault-aware,utility-based job scheduling on Blue Gene/P systems, Proc. IEEE Int'l Conf. Cluster Computing, pp.1-10(2009)