Vol.2016-HPC-155 No.12
2016/8/8

ネットワークトポロジーを考慮したノード故障状況の評価手法の検討

宇野 篤也[1,a)]　関澤 龍一[2]

概要：近年，スーパーコンピュータや PC クラスタといった HPC システムの高並列化により構成部品数が増加し，システムの故障率が高くなる傾向にある．通常の運用では，数台のノード故障がシステムの運用停止を引き起こすような事態はほとんど発生しないが，ジョブスケジューリングの観点からみると数台のノード故障でも故障の発生場所によっては運用へ大きな影響がでることがある．故障発生時に直ちに保守を実施することで故障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などの面から難しい．そこで我々は，ノードの故障状況にもとづいて保守タイミングを決めることで，システム利用率を大きく低下させることなく保守の実施回数を減らす手法を提案している．これまでは 1 次元のネットワークを対象としていたが，今回，多次元ネットワーク下での評価手法について検討を行ったので報告する．

1. はじめに

近年，スーパーコンピュータや PC クラスタといった HPC システムは高並列化の傾向にあり [1]，システムの構成部品数も増加している．例えば，理化学研究所 計算科学研究機構 (AICS) が運用を行なっている「京」では，システムを構成する部品数は 100 万点以上にもなる．構成部品数の増加は，システムの故障率が高くなることを意味する．

通常の運用では，数台の計算ノードの故障がシステムの運用停止を引き起こすような事態になることはほとんどないが，ジョブスケジューリングの観点からみると数台の故障でも故障ノードの発生場所によっては運用へ大きな影響がでる場合がある．故障発生時に直ちに保守を実施することでそ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などの面から難しい．そこで我々は，ノード故障状況にもとづいて保守を実施するタイミングを決定し，故障ノードの発生が運用に及ぼす影響を最小限にしつつ，保守作業の回数を減らす保守タイミングを決定する手法を提案した [2]．これまでは 1 次元のネットワークを対象としていたが，今回，多次元のネットワークにおける評価手法について検討を行った．また，ノード故障がシステムの利用率へ及ぼす影響の評価方法についても検討を行ったので報告する．

1　国立研究開発法人理化学研究所 計算科学研究機構
2　富士通株式会社
a)　uno@riken.jp

2. ノード故障とジョブスケジューリング

HPC システムの多くはバッチ形式でジョブを実行している．ジョブスケジューラは，投入されたジョブを計算ノードの利用状況に応じて，システムの利用率が最適になるように計算ノードをジョブに割り当てる．この時，ディスクの使用状況や計算ノード間のネットワーク構成等を考慮して割り当てる計算ノードを決定する．特に，直接網のネットワークをもつシステムでは，通信性能を確保するために隣接した計算ノード群を割り当てる場合がある．そのため，ノード故障が発生した場合，連続して計算ノードを確保できる空間が分断され，スケジューリングに大きく影響を及ぼす場合がある．

図 1 にノード故障が発生した場合のスケジューリング空間の例を示す．図 1 からわかるように，同じ故障数でも発生場所によって連続して確保できる空間に大きく差が生じ，単純な故障数で評価することは適切ではないことがわかる．

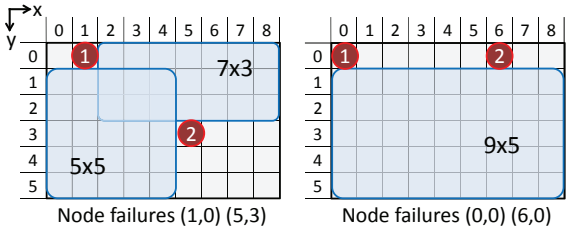

図 1　ノード故障とスケジューリング空間の関係

そこで我々は，故障状況を評価する方法として，ノード故障によって分割されたスケジューリング空間に着目し，システム全体のスケジューリング可能な組み合わせにもとづく評価式を提案した [2].

3. 故障状況の評価

3.1 スケジューリング可能な組み合わせにもとづく評価

この評価方法では，スケジューリング空間においてジョブがスケジューリング可能な組み合わせ数をもとに評価値を求める．スケジューリング空間はノード故障により分割された全ての空間を対象にする．評価式 E_c を式 (1) に示す.

$$E_n = \sum_{i=j}^{k} {}_{S_n-i+1}C_1 \qquad E_c = \frac{\sum_{i=0}^{k} E_i}{E_{max}} \qquad (1)$$

ここで，E_n はノード故障により分割されたスケジューリング空間におけるサイズ j〜k のジョブがスケジューリング可能な組み合わせの総和を，E_{max} はノード故障が発生していない状態での E_n を，S_n はスケジューリング空間の大きさ（例えば 1 次元の場合はノード数に一致）をそれぞれ表している．なお，$S_n < i$ の時はスケジューリングできないので $E_n = 0$ とする.

1 次元のネットワークの場合，スケジューリング可能な組み合わせ数は式 (1) のように単純な組み合わせで表現することが可能である．しかし，多次元のネットワークの場合はスケジューリング可能な組み合わせを表現することは難しい．そこで，スケジューリング空間の各点でのスケジューリングの可否を判断し評価することにした.

3.2 スケジューリング空間の各点におけるスケジューリングの可否にもとづく評価

この評価方法では，スケジューリング空間の各点でのスケジューリングの可否にもとづいて評価する．スケジューリングの可否とは，ある評価点においてジョブが要求するノード数を確保できるかという意味である．重複評価を避けるため，評価点を基準に各軸の正方向の空間に対してスケジューリング可能かで判断する．**図 2** に 2 次元のネットワークでのスケジューリング可否の判定例を示す．評価点（C）を起点とし，探索範囲（緑の領域）にノード故障（F）

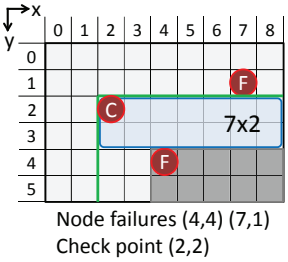

図 2 スケジューリング可否の判定例

があった場合，ノード故障を起点とし各軸の正方向の空間をスケジューリング不可（黒の領域）とする．その後，スケジューリング可能な空間（白の領域）と評価点を頂点とする矩形の面積を求め（この例では 14），要求するノード数より大きければスケジューリング可能と判断する．これを全ての評価点について行う．今回は簡単のため，ジョブ毎に指定された次元は考慮せずノード数のみで評価する．ジョブの次元指定を考慮する場合は，各評価点で指定された空間がスケジューリング可能かで判定する.

式 (1) では対象とする全てのジョブのスケジューリング可能な組み合わせ数で評価したため，特定のサイズのジョブの組み合わせ数が多くなると評価値に偏りがでるという問題点があった．そこで，ここではジョブのサイズ毎に評価値を求め，複数のサイズのジョブを対象とする場合はその平均値とする.

評価式 E_p を式 (2) に示す.

$$E_n = \frac{P(n)}{P(n)_{max}} \qquad E_p = \frac{\sum_{i=j}^{k} E_i}{k-j+1} \qquad (2)$$

ここで，$P(n)$ はサイズ n のジョブがノード故障により分割されたスケジューリング空間におけるスケジューリング可能な点の総数を，$P(n)_{max}$ はノード故障が発生していない状態での $P(n)$ をそれぞれ表している.

3.3 評価式の評価

ここで提案する評価式とスケジューリング空間の大きさの関係について評価を行った．式 (2) からわかるように，評価するジョブのサイズが評価値に大きく影響する．そこで，今回の評価では後述するシミュレーションで用いたジョブミックスをもとにジョブの規模別に 10 個のグループを作成した．表 1 にジョブ数およびジョブが消費した計算資源量で分類した結果を示す．ここでは，10–90%の各区切りにおけるジョブサイズを示している．表 1 からわかるように，ジョブ数で分類した場合は顕著な偏りがみられるため，ジョブが消費した計算資源量で分類することにした.

評価はネットワークトポロジーが

(1) 3 次元メッシュの場合

(2) 「京」と同じく x 軸をトーラス，yz 軸をメッシュとした 3 次元メッシュ・トーラスの場合

について実施した．空間分割は yz 軸の分割で行い，(2) の場合の x 軸のトーラスは維持されたままとする．評価に用いたスケジューリング空間は「京」と同じ $24 \times 18 \times 16$ とした.

図 3，**図 4** にスケジューリング空間の分割状況と各評価値の関係を示す．図 3 は分割されたスケジューリング空間

表 1 グループ分類結果 (ノード数)

分類方法	10%	20%	30%	40%	50%	60%	70%	80%	90%
ジョブ数	396	408	504	516	648	804	1,032	2,040	2,568
計算資源量	516	768	1,032	2,016	2,400	3,840	4,812	8,196	15,360

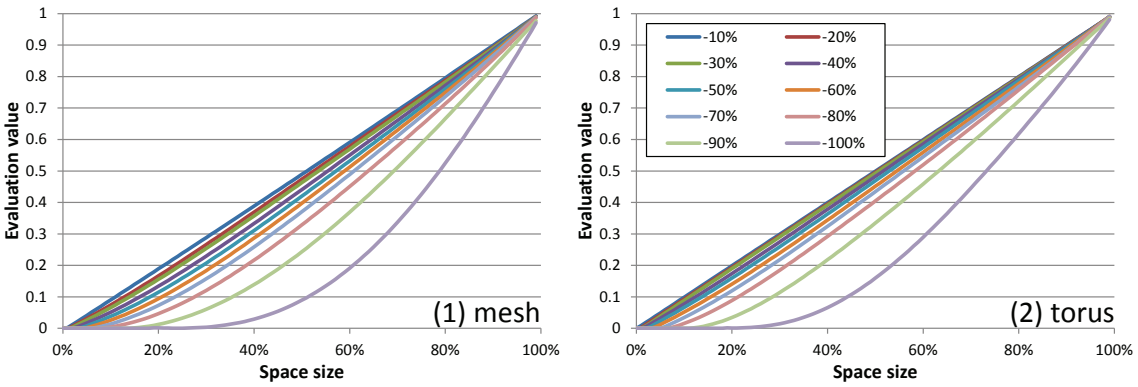

図 3 スケジューリング空間と評価値の関係

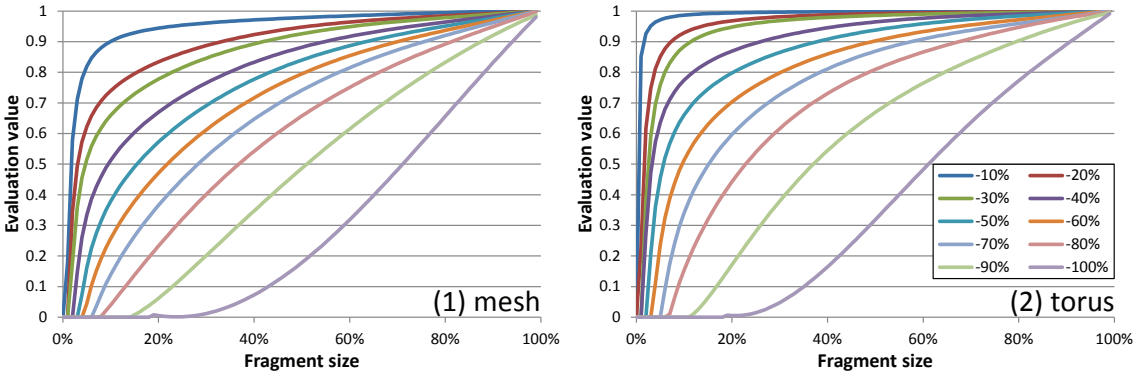

図 4 等分割した場合のスケジューリング空間と評価値の関係

毎の評価値を，図 4 はスケジューリング空間全体を等分割した場合のシステム全体の評価値（全ての分割されたスケジューリング空間を対象にした評価値）をそれぞれ示している．これらの図からわかるように，ジョブの規模に応じて評価値が低くなっている．10%から 80%までのグループはあまり差がみられないが，90%以上のグループは評価値が急激に低くなる傾向がみられる．

(1) と (2) を比較してみると，(2) の場合の方が評価値の減少が緩やかになっている．これは，今回の評価では x 軸のトーラスが維持されているため，(1) と比較して x 軸方向の空間を大きくとることができるためである．また，トーラス構造では 1 ノードの故障ではメッシュとなりスケジューリング空間は分割されないため，スケジューリングへの影響は小さい [3].

3.4 シミュレーションによる評価

ここで提案する評価式に基づいて保守タイミングを決定した場合のシミュレーションを行った．E_p は，図 4 の評価結果から規模の小さいグループではあまり差が出ないことが予想されたため，規模の大きいグループのグループ 1（70%–90%）とグループ 2（80%–100%）の評価値を使用した．また，保守タイミングはそれぞれの評価値が，60%，80%以下になった場合に保守を実施する条件で評価した．なお，保守の実施時刻はノード故障の発生時刻に関係なく，シミュレーション内時間での評価値の条件を満たす故障が発生した翌日の 0 時とした．

3.4.1 評価環境

今回の評価では，「京」で使用されているジョブスケジューラのシミュレータを使用した．シミュレータは，あらかじめ作成したジョブミックスとノード故障パターンを読み込み，各ジョブに設定された投入時刻に従って，ジョブの実行処理をシミュレートする．また，ノード故障パターンに従って，ノード故障を発生させる．この時，実行中のジョブがあった場合には当該ジョブの実行を中止し，故障ノードをスケジューリングの対象外とする．実行が中断されたジョブは再スケジューリングとなるが，再実行時に優先度を上げる等の処置は実施していない．なお，除外された計算ノードは保守実施後にスケジューリング対象となる．

今回使用したスケジューリングアルゴリズムは FCFS[*1] と Backfill[*2] である．「京」のノード数は 82,944 台だが，「京」ではスケジューリングは Tofu 単位で行なわれるため，実際のスケジューリング空間は $24 \times 18 \times 16$ となる．なお，今回のシミュレーションではステージングは実施しないものとした．

使用したジョブミックスは「京」で実行されたジョブの

[*1] First-Come and First-Served：ジョブの投入順に優先順位を決定するアルゴリズム
[*2] Backfill：空ノードがある場合にジョブの実行順序を入れ替えてシステム利用率を改善するアルゴリズム

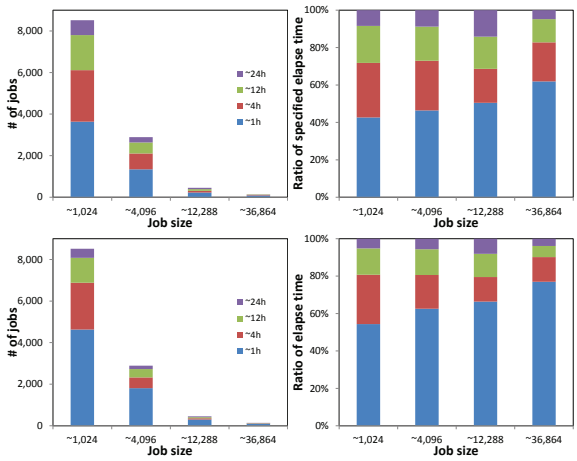

図 5 シミュレーションで使用したジョブセットの統計情報

統計情報をもとに生成した [4]．**図 5** に今回使用したジョブミックスの統計情報を示す．ジョブの最大ノード数は 36,864 ノード，ユーザがジョブを投入する際に指定する指定経過時間は最長で 24 時間とした．図 5 からわかるように，規模の小さいジョブの割合が多く，指定経過時間と実際の経過時間の差が大きいという特徴がある．ジョブミックスは，ジョブ投入期間を 28 日間（4 週間），密度を 85%として生成した．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全ジョブの実際に計算ノードを使用したノード時間積の総和の割合である．つまり，85%の密度をもったジョブミックスでシミュレーションを実行した場合，システム利用率は最大でも 85%となる．

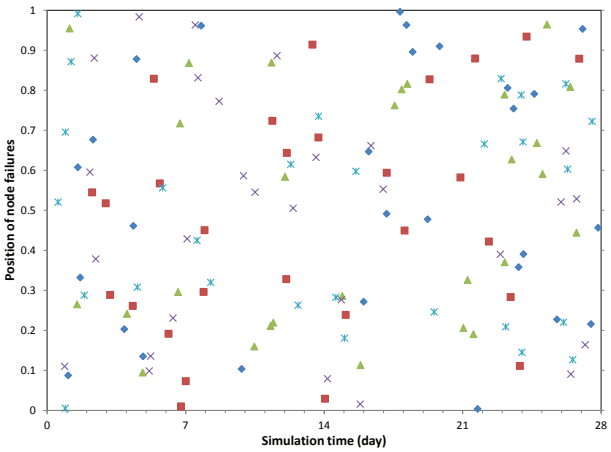

図 6 ノード故障パターン（5 セット分）

図 6 にシミュレーションで使用したノード故障パターンを示す．縦軸がノード故障が発生した個所（全ノード数を 1 として計算した場合の位置）を，横軸が発生日時を表している．5 セット分を示しているため，5 種類の記号がプロットされている．ジョブ投入期間中に 1 日に約 1 件のノード故障が発生する確率で生成しており，複数回ノード故障が発生する日や，一回も発生しない日もある．

ジョブミックスは 5 セットを，ノード故障パターンは 5 セットをそれぞれ作成した．今回の評価では，これらジョブミックスとノード故障パターンを組み合わせ，1 条件に対し $5 \times 5 = 25$ パターンのシミュレーションを実施した．ジョブ実行に関しては集計期間内に実行が開始されたジョブの実行待ち時間とジョブ全体に対する実行割合を，システムの運用効率に関してはシステム利用率と保守の実施回数を比較している．集計期間はシミュレーション開始後 2 日目からジョブ投入が終了する 28 日目までの 27 日間とした．最初の 1 日目を除外したのは，シミュレーション開始直後は投入されたジョブが少なく評価に適さないためである．なお，システム利用率の計算では，ジョブの実行中にノード障害が発生した場合は，ジョブ実行開始からノード障害発生までの間はその計算ノードは使われていなかったものとして計算している．

3.4.2 シミュレーション結果

図 7 にシステム利用率を，**図 8** に保守の平均実行回数と平均保守間隔を，**図 9** にジョブの実行待ち時間とジョブの終了割合をそれぞれ示す．

システム利用率には大きな差は見られなかった．これはジョブ密度が低いためと思われる．ジョブ密度 85%は「京」の実行状況を基に決定した値であるが，今回の評価ではステージングは実施しないため，スケジューリング効率が上がっているものと考えれらる．一方，保守の平均実行回数はグループ 1，グループ 2 とも大幅に削減できていることがわかる．また，ジョブの実行待ち時間とジョブの終了割合を見てみると，グループについては，グループ 1 > グループ 2 の関係が，評価値については $E = 0.8$ の場合 > $E = 0.6$ の場合 となっており，これらの結果は 1 次元ネットワークの場合と同じ結果となっている [2]．

以上の結果から，今回提案する評価式は多次元のネットワーク環境下で正しく評価できていると考えられる．

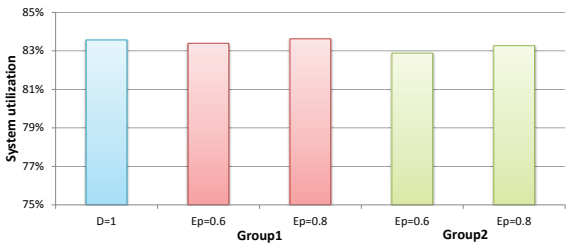

図 7 システム利用率

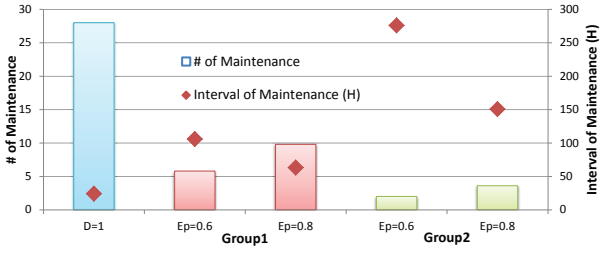

図 8 保守回数と保守間隔

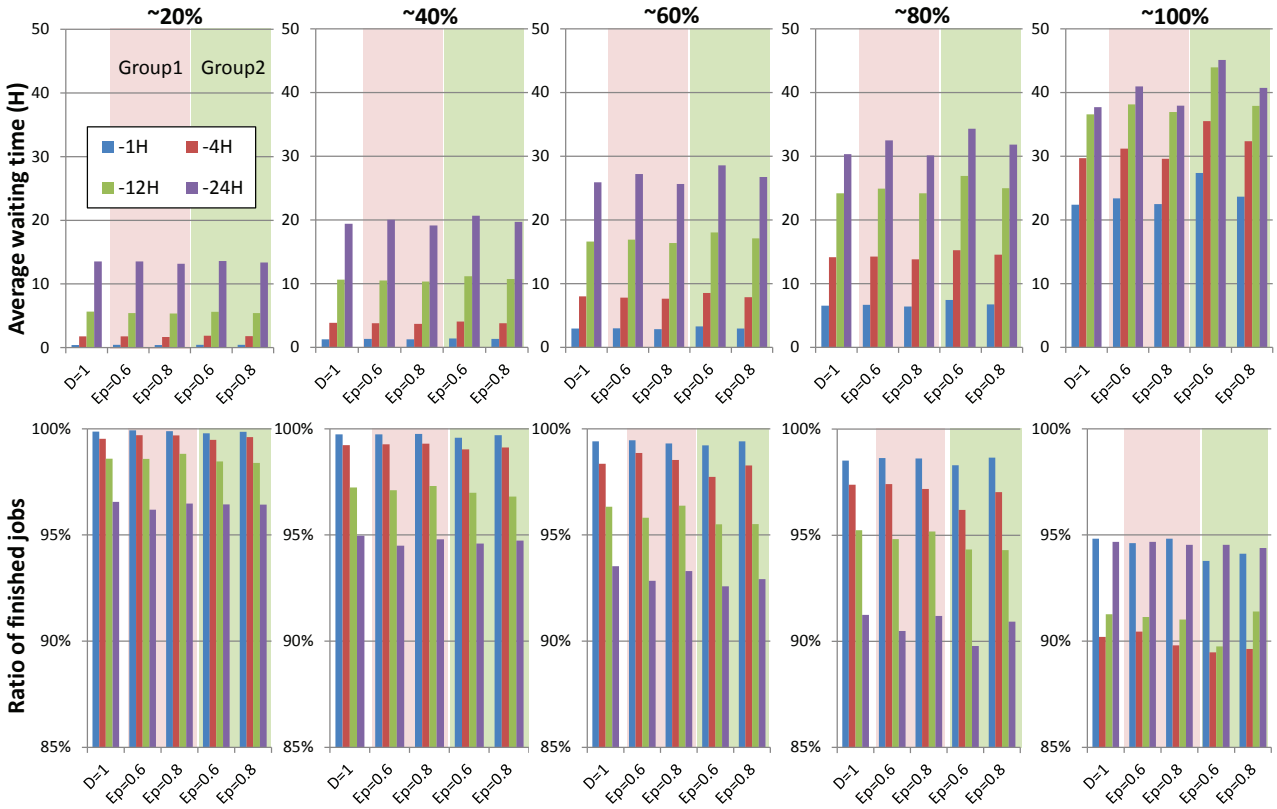

図 9 ジョブの実行待ち時間とジョブの終了割合

表 2 投入資源量と無故障時の消費資源量（Ro）

グループ	投入資源量	Ro
A	4	3
B	2.5	2
C	1.5	1

4. ノード故障のシステム利用率への影響の推測

次に，評価式を利用してノード故障のシステム利用率への影響の推測を行った．今回提案した評価式はノード故障によるスケジューリング可能な領域の変化率を表している．そこで，この評価値を利用してノード故障によるシステム利用率への影響を求められないか検討を行った．

評価値 E をとるときの，グループ i が消費した単位時間あたりの計算資源量 $R_i(E)$ を以下で定義する．

i が最大規模のグループの場合：$R_i(E) = E \times Ro_i$

i が上記以外のグループの場合：$R_i(E) = E \times remain \times r(i)$

ここで，Ro_i は無故障時にグループ i が消費した計算資源量を，$remain$ はその時点での残処理可能計算資源量を，$r(i)$ は無故障時のグループ i が消費する計算資源量の割合をそれぞれ示す．ただし，$R_n(E)$ が投入ジョブの資源量を上回る場合，投入ジョブの資源量が各グループで消費される資源量とする．

図 10 にシステム利用率の推測例を示す．各グループの投入された計算資源量および無故障時に消費された計

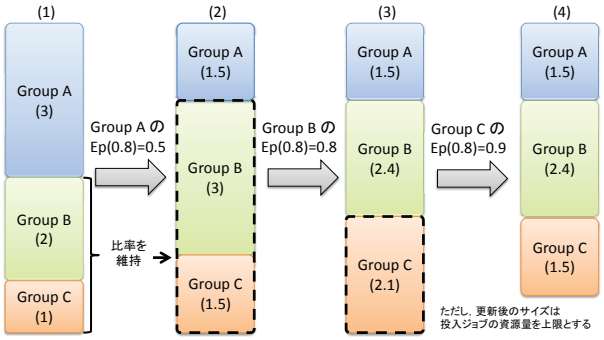

図 10 システム利用率の推測例

算資源量 Ro を表 2 とし，基準となるある評価値をとるときの F（図 4 の横軸）を 0.8 とする．また各グループに属するジョブのサイズは $A > B > C$ の順とする．$F = 0.8$ の時のグループ A の評価値が $E_p(0.8) = 0.5$ をとる場合，グループ A の故障時の消費資源量は $R_A(0.8) = E_p(0.8) \times Ro_A = 0.5 \times 3 = 1.5$ となる．この時，残りの計算資源量 4.5 はグループ B とグループ C で消費され，各グループ間の比率は無故障時の比率を維持するものと仮定する（図 10 の（2））．$F = 0.8$ の時にグループ B の評価値が $E_p(0.8) = 0.8$ とすると，グループ B の故障時の消費資源量は $R_B(0.5) = E_p(0.8) \times remain \times 2/3 = 0.8 \times 4.5 \times 2/3 = 2.4$ となり，残計算資源量は 2.1 となる（図 10 の（3））．$F = 0.8$ の時のグループ C の評価値が $E_p(0.8) = 0.9$ をとるとすると，グループ C の故障時の消費資源量は

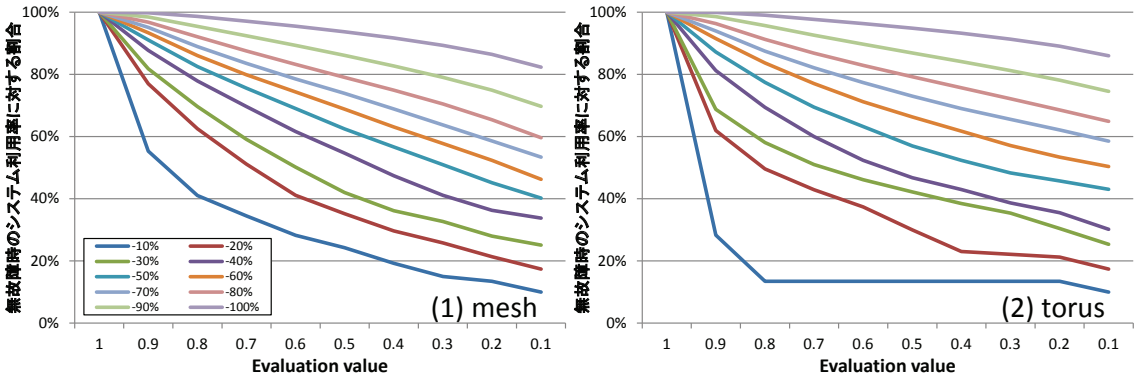

図 11 評価値に基づいて推測したシステム利用率の変化

表 3 システム利用率と推定値

グループ (評価値)	システム利用率	推定値	差
グループ 1 (0.6)	83.39%	83.19%	-0.20%
グループ 1 (0.8)	83.62%	84.07%	0.45%
グループ 2 (0.6)	82.88%	80.01%	-2.87%
グループ 2 (0.8)	83.27%	82.89%	-0.38%

$R_C(0.5) = E_p(0.8) \times remain \times 1/3 = 0.9 \times 2.1 = 1.89$ となるが，グループ C の上限値を超えるため，$R_C(0.5) = 1.5$ となる (図 10 の（4)).

図 11 に今回のシミュレーション結果をもとに，各グループの評価値と前述のアルゴリズムに基づいて計算したシステム利用率の推測値の関係を示す．図 11 からわかるように，規模の小さなグループで評価した場合，評価値が大きい段階でもシステム利用率は大きく低下することが予想される．逆に，大きいグループで評価した場合，評価値は小さくてもシステム利用率はあまり低下していない．しかし，スケジューリング空間は規模の大きいグループからみて小さく分割されているため，規模の大きいグループの待ち時間は長期化することが予想される．

図 11 では評価値は一定という条件で評価を行ったが，実際の運用では評価値は変動するため，実際の運用の場合と比較して推定されるシステム利用率は低くなる傾向にある．そこで，今回のシミュレーションで使用したノード故障パターンにもとづき，時系列で変化する評価値を用いてシステム利用率の推定を行った．表 3 に結果を示す．今回のシミュレーション結果との比較では，最大で 3%程度の誤差であった．

5. おわりに

本稿では，多次元ネットワーク環境における故障ノードの発生状況を評価する手法について述べた．本手法では，スケジューリング空間の各点におけるスケジューリング可能空間の大きさに基づいて評価値を決定する．各評価点毎に空間を計算するため，計算コストは高くなるが様々な条件下で評価が可能である．シミュレーションによる評価では，1 次元ネットワークの環境での評価結果と同様の結果を得ることが確認できた．また，この評価式を用いてノード故障のシステム利用率への影響評価を行った．今回のシミュレーション条件下では，最大で 3%程度の誤差でノード故障発生時のシステム利用率を求めることができた．

今回，多次元のネットワーク環境下での評価を行ったが，これは静的な環境での評価である．さらなる最適化を行うためには，投入されているジョブの状況を考慮し，評価を行う必要があると考えている．また，ノード故障のシステム利用率への影響推測では，今回は評価式を単純なスケジューリング可能な領域の変化率とみなして評価したが，さらなる検討が必要と考えている．

参考文献

[1] TOP500: TOP500 Supercomputer Sites, Top500.org (http://www.top500.org).
[2] 宇野篤也，関澤 龍一：ノード保守タイミングのジョブスケジューリングへの影響評価，ハイパフォーマンスコンピューティングと計算科学シンポジウム (2016).
[3] 関澤 龍一，宇野篤也，山本啓二，若林大輔，庄司文由：ジョブスケジューリングにおけるスケジューリング空間の評価，情報処理学会研究会報告 Vol.2015-HPC-149 No.1 (2015).
[4] 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).