Vol.2019-HPC-168 No.23
2019/3/7

スーパーコンピュータ「京」における ログデータに基づいたファイルステージングの分析と評価

伊藤 俊[1,a)]　山本 啓二[1]　松田 元彦[1]　辻田 祐一[1]　庄司 文由[1,b)]

概要：スーパーコンピュータ「京」（以下「京」）では，計算ノードのファイル I/O 性能を確保するために 2 階層のファイルシステムを持っている．計算ジョブを実行するためには，これらのファイルシステム間でファイルステージング（ファイルのコピー）を行う必要がある．「京」では，一定の制約の下で，ジョブの実行と平行してファイルステージングを行うことが可能であり，うまく動作すれば，ファイルステージング時間を隠蔽することが期待できる．本稿では，2 年分の「京」のログデータに基づき，計算ノードへの転送（ステージイン）時間の隠蔽について分析・評価を行った．分析の結果，ジョブ単位で見た場合，12,289 から 36,864 までのノード数規模において 80%を超えるジョブが隠蔽できていることが判明した．また，時間で見た場合，全ジョブの合計でステージイン時間の約 55.1%が隠蔽できていることが判明した．一方で，「京」の計算ジョブの大半はジョブの実行時間と比較してステージイン時間が短く，そのようなジョブにおいては，隠蔽によって得られる時間短縮効果が低いことも明らかになった．

1. はじめに

「京」は理化学研究所と富士通株式会社が共同で開発し，2012 年より共用を開始した [1][2]．「京」は，計算ノードの実効的なファイル I/O 性能を確保するために，相対的に大容量かつ狭帯域のグローバルファイルシステム（以下，GFS）と，小容量かつ広帯域のローカルファイルシステム（以下，LFS）の 2 階層のファイルシステムを持っている [3]．ユーザーファイルは GFS に保存されるが，ジョブ実行に必要なファイルは実行前に GFS から LFS にコピーされ，出力ファイルは実行終了後，LFS から GFS にコピーされる．これにより，実行中のジョブのファイル I/O は広帯域の LFS との間に閉じることになるので，実効的に高い I/O 性能が期待できる．しかし，ナイーブな実装だと，上記のファイル転送の間，CPU は待機状態になるので，システムの利用効率が低下するという問題がある．そこで「京」では，ノード利用率の向上のために，これらのファイル転送がジョブ実行とオーバーラップして実行できるように設計した．その一方でこの方式は，オーバーラップがない場合と比較して，実装やシステム構成が複雑になることで障害が起きやすくなったり，設定パラメタが多岐に渡るため，最適化が難しくなるなどの，運用面での困難も想定される．そのため，オーバーラップがある場合の効果や影響を検証することは重要である．

本研究の目的は，「京」で実行された膨大なジョブの履歴を分析し，「京」におけるジョブ実行前後のファイル転送方式の有効性を定量的に評価することである．本稿では，その最初のステップとして，「京」で実行されたジョブのファイル転送に関する特性を分析するとともに，ファイル転送とジョブ実行のオーバーラップの状況の評価に関する予備的な調査を実施した．

以下，2 章で，「京」におけるファイルステージングの方式を説明し，3 章で，これまでに「京」で実行されたジョブの特性について，ファイルステージングの観点での分析を行い，4 章で，今回実施したステージインとジョブ実行のオーバーラップの状況の予備的な評価について報告する．

2. 「京」のファイルステージング

2.1 一般論としてのファイルステージング

「京」のような大規模 HPC システムにおけるジョブの実行時間は主に，演算処理，ノード間の通信処理，ファイル I/O 処理から成る．このうち，ファイル I/O 処理については，ストレージが 1 階層しかない場合，さまざまな I/O リクエストが競合し，処理に時間がかかることがある．これを回避するためのひとつの方法として，ストレージを 2 つに分け，ジョブ実行中の I/O リクエストを 1 階層目で，それ以外のリクエストを 2 階層目で，それぞれ分けて処理する方法がある．これにより，競合が緩和あるいは回避でき

[1] 理化学研究所 計算科学研究センター
[a)] shun.ito@riken.jp
[b)] shoji@riken.jp

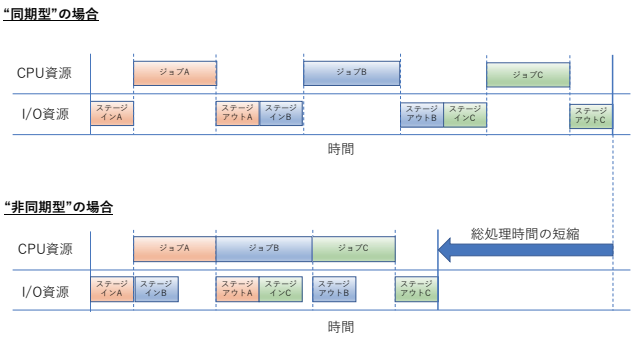

図 1　ファイルステージング方式の違い

ることから，ジョブ実行中の I/O 性能の向上が期待できる．その際，ジョブ実行の前後に 2 つのストレージ間でファイルの転送を行うことをファイルステージングと呼び，特にジョブ実行の前に 2 階層目から 1 階層目にファイルを転送することをステージイン，ジョブ終了後に 1 階層目から 2 階層目に転送することをステージアウトと呼ぶ．

次に，ジョブ実行とファイルステージングのスケジューリングについて考える．

最もシンプルな実装は，ステージイン，ジョブ実行，ステージアウトを 1 セットと見て，セット単位でスケジューリングする方法である．この場合，当該ジョブの実行開始時刻が来たら，ステージイン，ジョブ実行，ステージアウトの順番で連続して処理されるため，ジョブスケジューラはセット単位でスケジューリングを行うだけで良くなる．本稿では，ステージング処理とジョブ実行が同期して動作するという意味で，この方式を “同期型” と呼ぶ．この方式のメリットは実装をシンプルにできる点であるが，ステージングの間，CPU が待機状態になるため，システムの利用効率を上げにくいというデメリットもある．

これに対し，ステージングとジョブ実行が独立かつ並行に行われる方式を “非同期型” と呼ぶことにする．図 1 に示すとおり，非同期型では，順番は維持しつつ，ジョブ実行と実行前のジョブのステージインや実行後のジョブのステージアウトが並行して処理されるため，うまく機能すれば，同期型で問題だった CPU の待機時間が減り，全体の処理時間を短縮することができる．

しかし，非同期型では，ステージイン，ジョブ実行，ステージアウトがそれぞれ独立にスケジューリングされることに加え，1 階層目のストレージ資源の容量を考慮する必要があるため，スケジューラの仕事が格段に複雑になる．さらに，ステージインの開始をジョブ実行のどれほど前に行うか，ジョブあたりの 1 階層目のストレージの利用量上限（クォータ）をどれくらいに設定するか等々，調整すべき設定パラメタが増えるため，最適化がより困難になるというデメリットもある．

2.2　京におけるファイルステージング

「京」におけるファイルステージング方式は，2.1 でいうところの”非同期型”である [4]．2.1 でも述べた通り，うまく機能すれば全体の処理時間を短縮することが可能であるが，実際には表 1 に示すとおり，いくつかの制約がある．

表 1　「京」のファイルステージング方式の制約

項目	制限
ジョブあたりのディスク使用量の上限	ノードあたり 14GB（既定）～ 100GB（最大）
多重度	ステージイン/ステージアウト合わせてノードあたり 1
ステージイン/ステージアウトの優先度	ステージアウトを優先

“非同期”にステージングとジョブ実行が行えるとしても，将来のジョブのステージインを先行して行う場合，第 1 階層のストレージ容量の上限を越えてはできない．ジョブが割り当てられたノードのうち，どれか一つでもストレージ容量が上限に達していたら，当該ジョブのステージインは保留される．また，I/O 資源の競合による転送時間の変動を抑えるため，同時に実行できるステージイン，ステージアウトは一つだけで，両者が競合した場合はステージアウトが優先される．

以上のような様々な制約の中で運用しているため，実際の運用では非同期型ステージングが効率良く機能できないケースも散見されている [5]．よって，分析・評価を通じて，これらの制約の効果や影響も含めた非同期型スケジューリングの有効性について評価を進めている．

3.　ジョブの特性分析

「京」で実行されたジョブのログデータに基づき，ジョブの特性について，ファイルステージングの観点で分析を行った．

3.1　分析対象ジョブ

ジョブの特性分析を行う際，対象としたジョブについて説明する．対象ジョブとした条件は以下の通りである．

- サンプリング期間：2016/10/1～2018/9/30
- ジョブキュー：small, large, huge
- ジョブの実行時間 (elapsed time) ＞ 600[sec]
- 正常終了したジョブ

まず，ジョブキューについては small，large，huge とし

表 2　「京」のジョブキュー

名称	ノード数	時間	ステージング	備考
small	1 ～ 384	24h	有り	
long	1 ～ 384	72h	有り	
micro	1 ～ 1152	0.5h	無し	
large	385 ～ 36864	24h	有り	
huge	36865 ～ 82944	8h	有り	3 日間/月

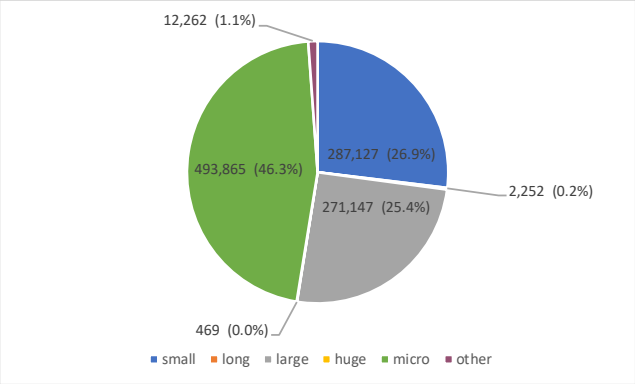

図 2　サンプリング期間におけるジョブ数

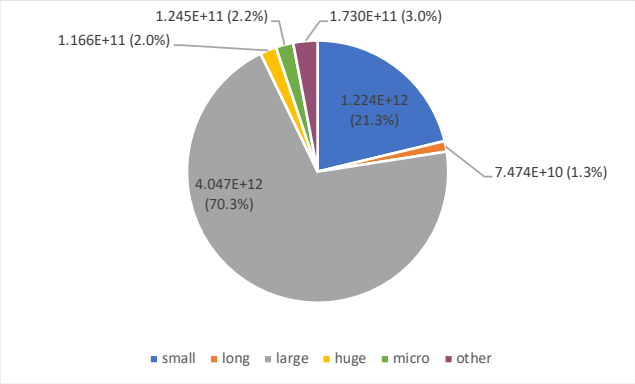

図 3　サンプリング期間におけるノード時間積 [秒]

た．ここで，「京」のジョブキューを表 2 に示す．加えて，サンプリング期間における「京」のジョブキュー毎のジョブ数を図 2，ノード時間積を図 3 にそれぞれ示す．ジョブの特性分析をファイルステージングの観点で行う目的であったため，ファイルステージングを行わないジョブキューである micro は除外した．long についてはファイルステージングを行うジョブキューであるものの，実行時間が 72 時間と他のジョブキューと比較して長く，ジョブ数・ノード時間積ともに全体と比較して 1.0%程度かそれ以下であったため除外した．

また，ジョブの実行時間（elapsed time）は 600 秒より大とした．理由はプロダクトランのジョブに限定して集計したかったためである．

上記条件によって抽出されたジョブの特性分析の対象となるジョブについて，ノード数規模毎に分類したジョブ数の内訳を図 4，ノード時間積の内訳を図 5 に示す．また，実行時間毎に分類したジョブ数の内訳を図 6，ステージイン時間毎に分類したジョブ数の内訳を図 7，ステージアウト時間毎に分類したジョブ数の内訳を図 8 にそれぞれ示す．

この期間に実行されたすべてのジョブ数に対する分析の対象となったジョブ数の割合を表 3 に示す．分析の対象となったジョブについて，ジョブ数で比較した場合にはこの期間に実行されたすべてのジョブの約 34.2%と少なく見えるが，ノード時間積で比較した場合にはこの期間に実行されたすべてのジョブの約 86.4%であり，これらのジョブに対して分析を進めても問題ないと考える．

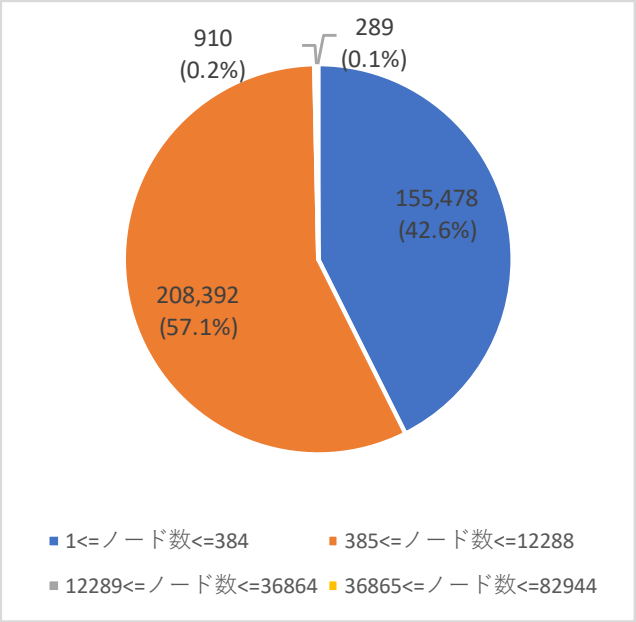

図 4　ジョブの特性分析対象のノード数規模毎のジョブ数の内訳

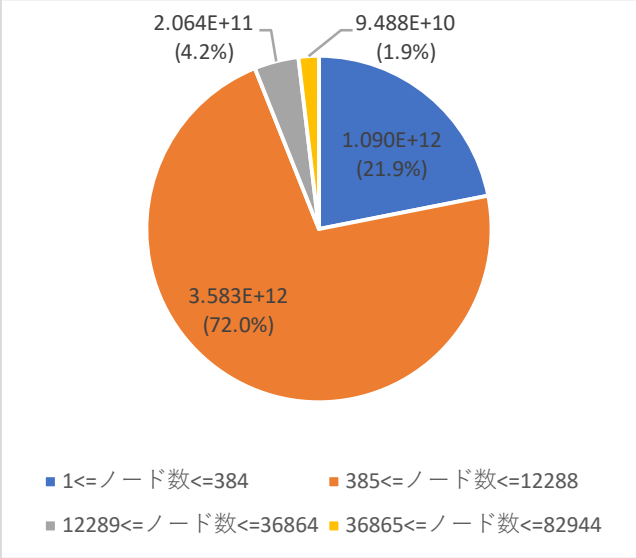

図 5　ジョブの特性分析対象のノード数規模毎のノード時間積 [秒] の内訳

表 3　分析対象ジョブの割合

	ジョブ数	ノード時間積 [秒]
分析対象期間の全ジョブ	1,067,122	5,760,087,642,107
分析対象ジョブ	365,069	4,974,502,189,541
分析対象ジョブ割合	34.211%	86.362 %

3.2　ステージング時間の分布

3.1 で抽出したジョブを用いて，ステージングにかかる時間について調査した．ジョブに割り当てられたノード数

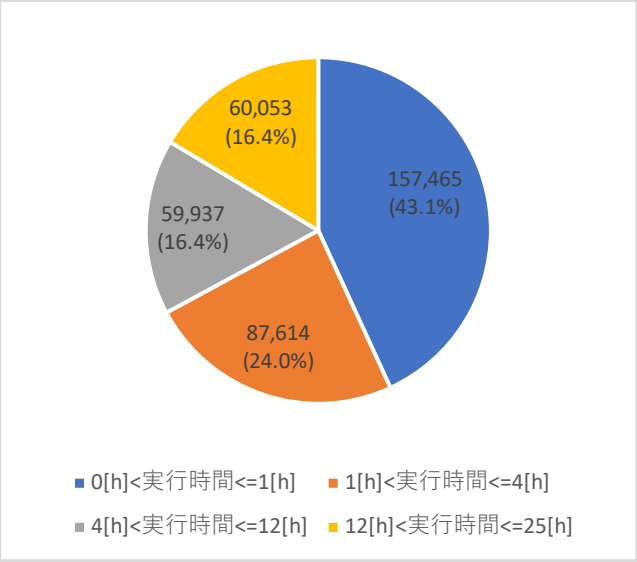

図 6　ジョブの特性分析対象の実行時間毎のジョブ数の内訳

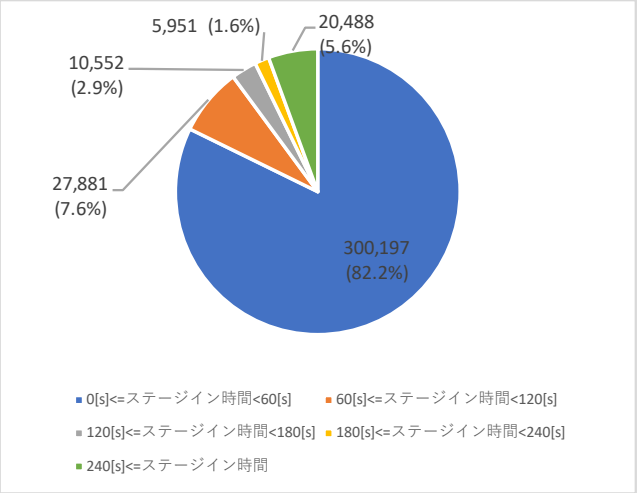

図 7　ジョブの特性分析対象のステージイン時間毎のジョブ数の内訳

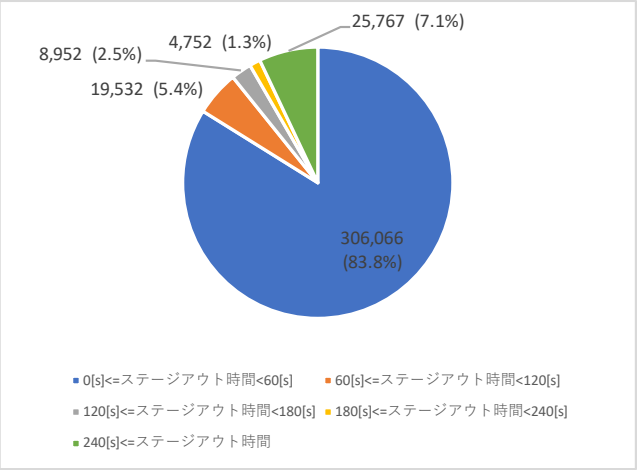

図 8　ジョブの特性分析対象のステージアウト時間毎のジョブ数の内訳

と，ジョブの実行時間でジョブを分類し，更にその項目毎にステージイン時間でジョブを分類した．分類した各項目

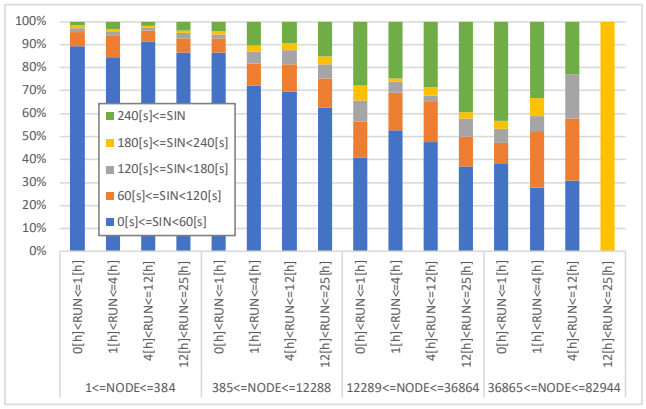

図 9　ステージイン時間毎のジョブ数の内訳

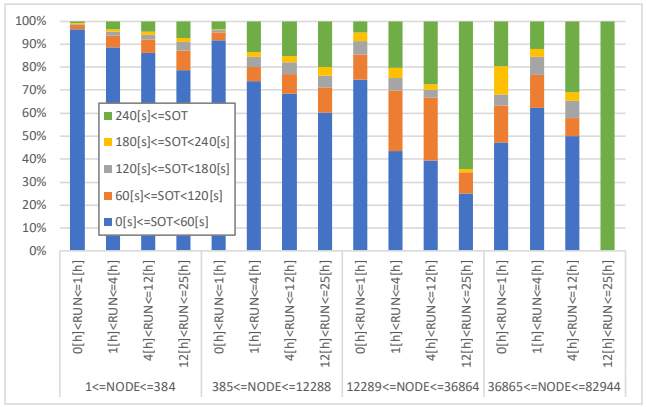

図 10　ステージアウト時間毎のジョブ数の内訳

におけるジョブ数の内訳を図 9 に示す．また，ステージアウトについても同様にジョブを分類した．ジョブ数の内訳を図 10 に示す．

ステージインについて確認すると，ノード数が 384 までのジョブについては，いずれの実行時間においてもステージイン時間の分布に変化はなかったが，ノード数が 385 以上のジョブについては，実行時間が大きくなるとステージイン時間も大きくなる傾向が確認できた．また，ノード数が 12,288 までのジョブについては，いずれの実行時間でも 80%のジョブはステージイン時間が 240 秒未満と小さい．一方，ノード数が 12,289 以上のジョブについては，各項目単位で見るとステージイン時間が 240 秒より大きいジョブの割合が大きく見えるが，全体で見ると図 4 に示す通りノード数が 12,289 以上のジョブは合計でも 1,199 であり少ない．全体のステージイン時間としては，図 7 に示す通りステージイン時間の短いジョブが大部分を占めており，ステージイン時間が 240 秒未満のジョブは全体の約 94.4%であった．また，ステージアウトについても，図 8 に示す通りステージインと同様の傾向であり，ステージアウト時間が 240 秒未満のジョブは全体の約 92.9%であった．

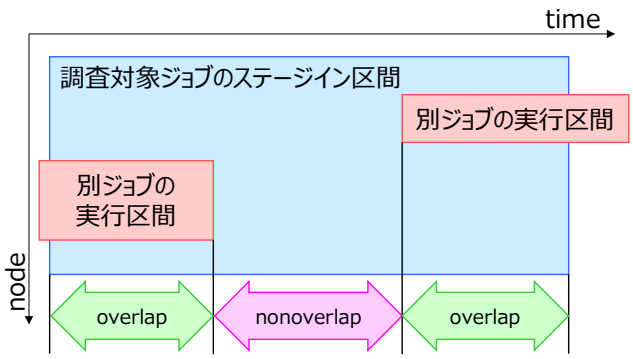

図 11　overlap と nonoverlap の定義

4. ステージイン時間の隠蔽の評価

4.1 ステージイン時間の隠蔽

ステージイン時間がどれほど隠蔽できたかジョブ毎に時間の単位で算出した．本稿において，隠蔽できた時間については overlap，隠蔽できなかった時間については nonoverlap と表現する．overlap と nonoverlap の定義を図 **11** に示す．「京」ではジョブのノード数と実行時間の要求に従って，計算ノードの空いている空間にスケジューリングされる．隠蔽できたか確認を行うジョブ（調査対象ジョブ）のステージインについては，調査対象ジョブの使用する計算ノードの一部，または全部が同じとなる別のジョブの実行時間とオーバーラップすることが可能である．調査対象ジョブのステージインを行っている区間の内，調査対象ジョブの使用する計算ノードの一部，または全部が同じとなる別のジョブの実行区間が最低でも 1 つ存在する区間については overlap，それ以外の区間を nonoverlap と，それぞれ定義する．これにより，調査対象ジョブのステージイン時間は，overlap か nonoverlap のいずれかに分類される．ジョブ毎に overlap と nonoverlap を算出し，集計を行った．

4.2 評価対象ジョブ

ステージイン時間の隠蔽の評価を行う際に対象としたジョブについて説明する．対象としたジョブは，3.1 で述べたジョブの特性分析で対象としたジョブの内，サンプリング期間以外の条件を除いたジョブである．ただし，ノード数規模については 1 ノードジョブを除外し，12 ノードから 82,944 ノードまでのジョブで結果を評価した．1 ノードジョブを除外した理由として，「京」では計算ノードにジョブを割り当てる際，基本的にはノード単位ではなく Tofu[6] 単位で割り当てを行うが，1 ノードジョブはノード単位での割り当てを行い，これが例外的な割り当てであるためである．

上記条件によって抽出されたステージイン時間の隠蔽の評価対象となるジョブについて，ジョブ数の内訳を図 **12**，ノード時間積の内訳を図 **13** に示す．全ノード数規模の合計値である 494,770 のジョブを対象として評価を行った．

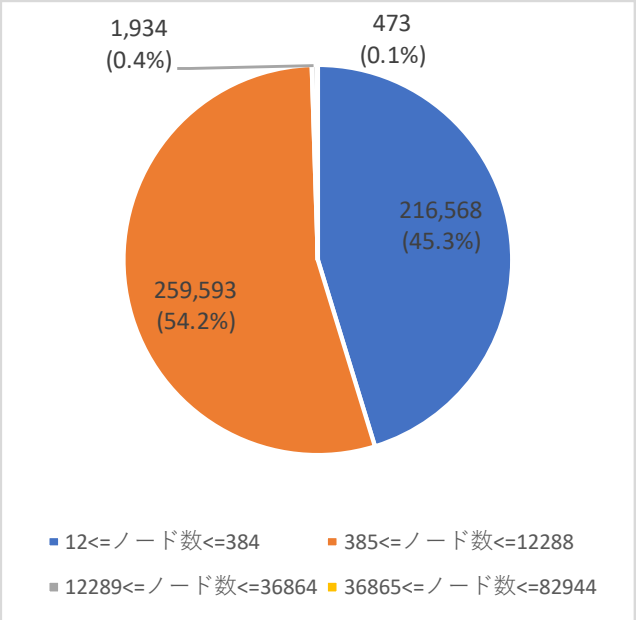

図 12　ステージイン時間隠蔽の評価対象ジョブのジョブ数の内訳

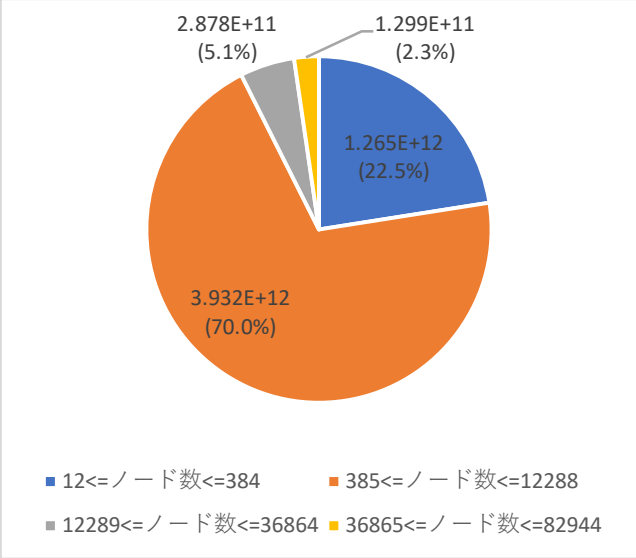

図 13　ステージイン時間隠蔽の評価対象ジョブのノード時間積の内訳

4.3 隠蔽率

ステージイン時間がどれほど隠蔽できたかを表す指標値として，隠蔽率を定義した．隠蔽率は，隠蔽できた時間と，ステージイン時間を用いて

$$\text{隠蔽率}\ (\alpha) = \frac{\text{隠蔽できた時間}}{\text{ステージイン時間}}$$

とした．

4.4 隠蔽の効果

ジョブ毎に隠蔽率を算出し，ノード数規模別のジョブ数の割合を図 **14** に示す．

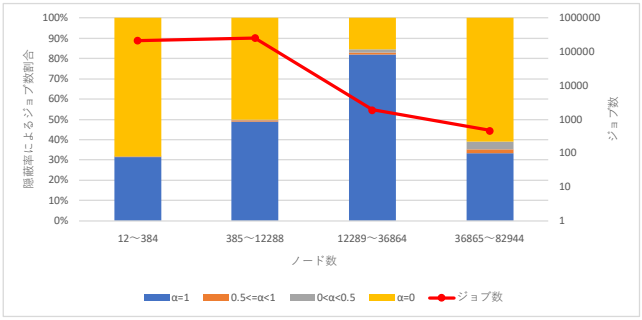

図 14 ノード数に対する隠蔽されたジョブ割合の変化

ジョブ数の割合を確認すると，36,864 ノードまでは隠蔽できたジョブの割合が右肩上がりに増加しており，ノード数が 12,289 から 36,864 までの規模では隠蔽できたジョブ数の割合が 80%を超える結果となった．これは，今回は隠蔽率を時間で評価しており，調査対象ジョブと同じ計算ノードを使用している別のジョブの実行区間が最低でも 1 つ存在すれば隠蔽できたジョブに分類されるため，使用するノード数が増えれば単純に別のジョブの実行区間と重なる可能性が高くなるからと考えられる．一方，36,865 から 82,944 までの規模では隠蔽できたジョブ数の割合が約 30%と低い．これは，ジョブ数が 473 本と，全体のジョブ数の 0.099%と少なく，サンプル数が不十分ではないかと考える．また，36,865 から 82,944 までの規模は，「京」のジョブキューでは huge に分類される．huge のジョブは月に 1 回実施される大規模ジョブ実行期間でしか実行されず，この期間の前には必ず保守が実施される．保守期間中はすべてのジョブ実行が行われないため，huge のジョブのステージインは他のジョブの実行の裏に隠れる可能性が低いと考えられる．従って，36,865 から 82,944 までの規模において，隠蔽できたジョブ数の割合が低くなっているのではないかと考える．

ジョブ数だけではなく，時間についても隠蔽の効果を確認した．ノード数規模毎に隠蔽時間の合計をステージイン時間の合計で割って，ステージイン時間に対する隠蔽時間の割合を算出した．その結果を**図 15** に示す．ステージイン時間に対する隠蔽時間の割合について，全ジョブの合計では，ステージイン時間の約 55.1%が隠蔽されているという結果で，ステージイン時間の半分以上は隠蔽できていることが確認できた．また，その内訳としてノード数規模毎に確認すると，隠蔽できたジョブ数の割合と同様の傾向であった．

5. まとめ

本稿では，「京」で採用したジョブ実行と非同期に実行できるステージングの効果について，同期型ステージングと比較してステージイン時間をどれほど隠蔽できたかという点で評価した．その結果，ジョブ数で見ると約 41.2%，時間で見ると約 55.1%の隠蔽効果があった．一方で，「京」

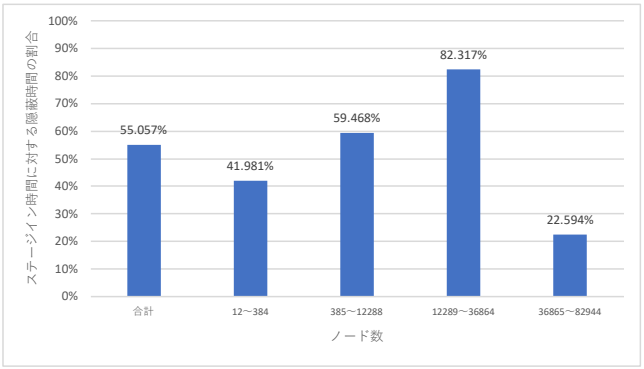

図 15 ノード数に対する隠蔽時間比率の変化

のジョブの大半は実行時間と比較してステージング時間が短く，そのようなジョブにおいては隠蔽によって得られる時間短縮効果が低いことが明らかとなった．また，ノード数規模で隠蔽率を比較した場合には，ノード数規模が大きくなるにつれて隠蔽率が高くなる傾向が確認できた．これは，4.4 でも述べたが，今回は隠蔽率を時間で評価しており，調査対象ジョブと同じ計算ノードを使用している別のジョブの実行区間が最低でも 1 つ存在すれば隠蔽できたジョブに分類されるため，使用するノード数が増えれば単純に別のジョブの実行区間と重なる可能性が高くなるからと考えられる．

今回は非同期型ステージングの効果について確認するため隠蔽率を時間で評価したが，ジョブの充填率という観点でノード時間積での評価も必要と考えている．これについては今後の課題と捉え，分析と評価を継続していくことを検討している．

参考文献

[1] Yokokawa, M., Shoji, F., Uno, A., Kurokawa, M. and Watanabe, T.: The K Computer: Japanese Next-generation Supercomputer Development Project, *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, Piscataway, NJ, USA, IEEE Press, pp. 371–372 (online), available from ⟨http://dl.acm.org/citation.cfm?id=2016802.2016889⟩ (2011).

[2] 特集：スーパーコンピュータ「京」，情報処理， Vol. 53, pp. 752–807 (2012).

[3] 酒井憲一郎，住元真司，黒川原佳：スーパーコンピュータ「京」の高性能・高信頼ファイルシステム，雑誌 FUJITSU, Vol. 63, pp. 280–286 (2012).

[4] 平井浩一，井口裕次，宇野篤也，黒川原佳：スーパーコンピュータ「京」の運用管理ソフトウェア，雑誌 FUJITSU, Vol. 63, pp. 287–292 (2012).

[5] Shoji, F.: Lessons learned from development and operation of the K computer, *Parallel Computing*, Vol. 64, pp. 12–19 (2017).

[6] 安島雄一郎，井上智宏，平本新哉，清水俊幸：スーパーコンピュータ「京」のインターコネクト Tofu，雑誌 FUJITSU, Vol. 63, pp. 260–264 (2012).