G-37

クラウドプラットフォームにおけるジョブスケジューリングの改善

Improvement of job scheduling in a cloud platform

萬代 光治†　　川原 純†　　笠原 正治†
Koji Mandai　　Jun Kawahara　Shoji Kasahara

1. はじめに

Google Cloud Platform (以下，GCP) や Amazon Web Service などに代表されるクラウドプラットフォームの需要は年々高まっており，需要に応じて規模が増大している．例えば GCP では 2016 年 11 月に東京リージョンの運用が開始された [1]．クラウドプラットフォームを用いる利点は，サービス利用料を支払えば，クライアントは計算機を購入・設置・保守運用することなく計算リソースを利用できることである．一方で，多くのユーザがプラットフォームを共有し，種類の異なる様々なジョブを実行させるため，ユーザ数が増えると，ジョブの実行が遅延しがちになる．遅延を避けるためには，ジョブへのリソース割り当てを緻密に行う必要がある．

Google はクラウドプラットフォームの研究を促すため，Google Cluster-Usage Traces（以下，トレースデータと呼ぶ）と呼ばれるデータを公開した [2]．このデータは，GCP 上でのジョブやタスクの実行状況を一定期間収集したデータであり，タスクが使用した CPU やメモリなどのリソースの相対的な量が記載されている．

Google Cluster-Usage Traces を利用した先行研究として，[3]では，GCP 上のマシンや投入されるジョブが様々な面で一様ではないことを明らかにし，ジョブに割り当てるスロット数や CPU 数を固定する従来のスケジューリング手法は非効率である可能性を示した．Liu ら [4] は，GCP を構成するマシンは数種類あり，それらが頻繁に故障やアップデートで使用不可になること，また実行に失敗したジョブやタスクに多くの CPU 時間が費やされていることなどを示した．[5] ではデータの解析を通して GCP と他社のプラットフォームサービスを比較し，GCP に投入されたジョブの多様性が他のサービスより高く，またジョブの到着頻度は他のサービスより高いが，変動が少ないことを示した．トレースデータはスケジューリングだけでなく，様々な解析に用いられている．[6] ではデータセンターの電力最小化問題に対し，主に需要変動やスケジューリング遅延のデータとしてトレースデータを用いている．

本稿では Google Cluster-Usage Traces を分析し，クラウドプラットフォームに投入されたジョブの性質と，計算リソースの使用状況を，ジョブが終了した原因ごとに明らかにする．そしてボトルネックとなるタスクを中断する方式を提案し，その効果をリソースの浪費の観点から評価する．

2. Google Cluster-Usage Traces

2.1　Google Cluster-Usage Traces の概要

Google クラスタは，高帯域幅のクラスタネットワークによって接続されたマシン群である．Google Cluster-Usage Traces は GCP の Web ページ [2] で公開されている． 2011 年 10 月に初めてバージョン 1.0 のデータが公開された．本稿では 2014 年 11 月に公開されたバージョン 2.1 のデータを利用している．

このトレースデータは，2011 年 5 月 2 日 08:00 (日本時間) から 5 月 30 日 09:00 までの Google クラスタプラットフォームのトレースデータであり，RPC の定期通信によって収集された． すべてのジョブとすべてのマシンには，一意の 64 ビット ID が割り当てられ，ID は再利用されない．

このデータでは機密保持のために，ジョブ名やユーザ名がランダムハッシュで匿名化された文字列になり，CPU コア数やメモリ容量は実際の値ではなく，0 から 1 の値に正規化された値が記録されている．例えば n コアのマシンのうち 1 コアを使用したとき，CPU コア数は 1/n と記録される．

データは 6 種類のテーブルで構成される．それらはジョブスケジューリングイベントを記録した Job Event テーブルとタスクのイベントを記録した Task Event テーブル，マシンのプラットフォームへの追加と削除を記録した Machine Event テーブル，各マシンのプロパティを記録した Machine Attributes テーブル，各タスクが使用するメモリや CPU の使用状況を記録した Task Usage テーブル，タスクが実行されるマシンを制限したときに記録される Task Constraints テーブルである．

2.2　スケジューリングイベントのデータ

Job Event テーブルはジョブのクラウドプラットフォームへの投入や，実行開始，実行終了などのイベントの発生とその時刻を記録したテーブルである．イベントのコードと種類を表 2.2.1 に示す．

表 2.2.1: イベントコードと意味

Event code	Event code が表す遷移
0	Submit (状態が UNSUBMITTED か DEAD から PENDING に遷移)
1	Schedule (状態が PENDING から RUNNING に遷移)
2	Evict (他のジョブ/タスクにリソースを与えるため停止)
3	Fail (ジョブ/タスクがエラーにより停止)
4	Finish (ジョブ/タスクが正常に終了した)
5	Kill (ジョブ/タスクがユーザまたはシステムのキルにより停止)
6	Lost (ジョブ/タスクが不明の理由で停止)
7	Update Pending (PENDING 状態のジョブ/タスクがマシンアップデート)
8	Update Running (RUNNNIG 状態のジョブ/タスクがマシンアップデート)

†奈良先端科学技術大学院大学，Nara Institute of Science and Technology

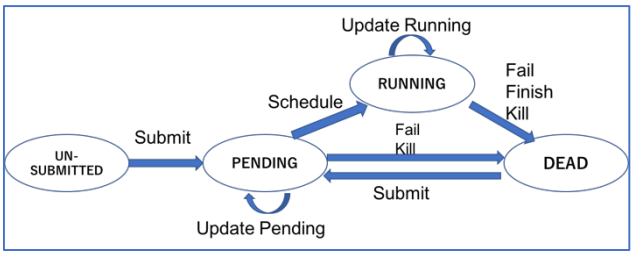

図 2.2.1: イベント状態と遷移の種類

ジョブは UNSUBMIT，PENDING，RUNNING，DEAD のいずれかの状態をとる．生成されたジョブは UNSUBMI-T 状態であり，ジョブが投入されると PENDING 状態に遷移する．ジョブの投入は Submit イベント（イベントコード 0）で表される．PENDING 状態のジョブは Schedule イベント（コード 1）によって RUNNING 状態に遷移する．その後，Finish （コード 4），Fail（コード 3），Kill（コード 5）イベントのいずれかによって，DEAD 状態に遷移する．Finish は正常終了，Fail はシステムエラーによる異常終了，Kill はユーザまたはシステムによる中断を意味する．Fail または Kill によって DEAD 状態に移った場合，Su-bmit イベントによって再び PENDING 状態に遷移することもある．これは失敗したジョブが再投入されたと考えられる．ジョブの状態遷移図を図 2.2.1 に示す．

ジョブは複数のタスクから構成される．各タスクに対しても，ジョブと同じイベントが発生し，同じ状態をとる．タスクに関するイベントは Task Event テーブルに記録されている．

2.3 マシンに関するデータ

Machine Event テーブルはプラットフォームを構成するマシンの追加や削除に加え，各マシンの CPU コア数やメモリ容量を記録している．Machine Attributes テーブルは，各マシンの，カーネルのバージョン，クロック速度，外部 IP アドレスの存在など，マシンのプロパティを表す．タスクが実行マシンを指定する際に用いられる．

2.4 計算リソースに関するデータ

Task Usage テーブルには，Linux containers を用いて，タスクが処理に利用した CPU コアやメモリ容量が記述されている．測定期間は最大 300 秒で，例えば 1 時間持続するタスクなら 3,600 [秒] / 300 [秒] = 12 回に分けてリソース利用状況が記録される．

3. 終了原因ごとのジョブ持続時間の解析

3.1 ジョブ持続時間

本節では，クラウドプラットフォームでどのようなジョブ・タスクが実行されているかを見るため，投入されたジョブの持続時間とタスク保持数の累積分布を示す．結果をそれぞれ図 3.1.1 と図 3.1.2 に示す．

図 3.1.1 は全てのジョブの終了時刻と開始時刻の差を持続時間とし，横軸の持続時間以下のジョブ数を加算してプロットした．クラスタに投入される持続時間は最短で 0.001 秒から，最長で 29 日以上と様々なものが混在している．平均値は 255 分，中央値は 3 分，分散は 2.83×10^{10} だった．図 3.1.2 は，横軸のタスク数以下で構成されるジョブ数を累積してプロットした．タスク保持数は最小で 1 個，最大で 90,050 個であり，様々なものが混在している．平均値は 37.9 個，分散が 1.07×10^{6} である．75%のジョブは 1 タスクから構成される．

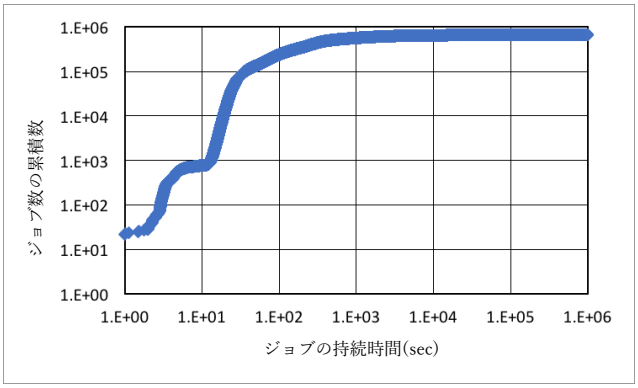

図 3.1.1: ジョブの持続時間の分布

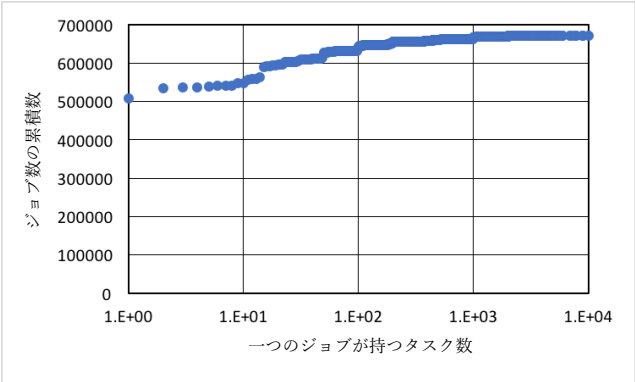

図 3.1.2: ジョブのタスク保持数の分布

次に，計算リソースを利用したジョブの中で，正常に終了しなかったジョブがどれだけ含まれているかを調べるため，ジョブのスケジューリング情報である Job Event テーブルの解析結果を示す．Job Event テーブルから ID ごとに発生したイベントを集計し，順に並べる．例えば ID383419 のジョブに対してイベントコードを時系列順に並べると，古い順から「014」となる．図 2.2.1 の状態遷移図と表 2.2.1 から，このジョブに対して，Submit，Schedule，Finish のイベントが発生し，正常終了したことが分かる．イベントコードの列が「015014」なら， Kill によって DEAD 状態に遷移したのち Submit イベントで PENDING 状態に戻り，その後で正常終了したことが分かる．イベントコード列ごとのジョブの数と割合を表 3.1.1 に示す．イベント列の大部分は 014, 013, 015, 01 の 4 通りであり，それぞれ，正常終了，エラーによる終了，ユーザまたはシステムによる中断（以下，キルと呼ぶ），実行中のジョブであると考えられる．実行中のジョブとは，データの集計期間の終了時に実行されているジョブであると考えられる．以上の分類を終了種類と呼び，それぞれのジョブを正常終了ジョブ，エラー終了ジョブ，キルジョブ，実行中ジョブと呼ぶ．特に，エラー終了ジョブとキルジョブをまとめて異常終了ジョブと呼ぶ．表 3.1.1 から，ジョブの大部分は DEAD 状態から Submit イベントで PENDING 状態に戻ることはないことが分かる．加えて，それぞれの終了種類のジョブが利用した CPU 時間とメモリの総量を表 3.1.2 に示す．各ジョブについて，ジョブの持続時間に，ジョブの平均 CPU 使用率（0 から 1 までの値をとる）と正規化された実行マシンコア数（0 から 1 までの値に正規化されている）を掛けた値を，ジョブの CPU 時間と定義する．

イベント列	終了状態	ジョブ数	ジョブ数の割合
014	正常終了	385581	57.375%
013	エラー	10124	1.506%
015	キル	272341	40.525%
01	実行中	3986	0.593%
その他		42	0.006%

表 3.1.1:全ジョブの終了パターンの割合

イベント列	終了状態	CPU 時間の割合
014	正常終了	0.710%
013	エラー	0.931%
015	キル	43.175%
01	実行中	54.377%

表 3.1.2:終了パターン別の CPU 占有率

表 3.1.1 から，投入されたジョブの 58%は正常終了ジョブであり，42%のジョブがエラージョブまたはキルジョブである．また，1%のジョブは DEAD 状態に移るイベントコードがなく，観測期間中の実行終了が確認できなかった．表 3.1.2 で示した結果は，CPU コア数は 2 章で述べたように正規化されており合計値は CPU 時間としては正確なものでないため，比率のみ表示している．この結果からは約 44%の CPU が正常終了しないジョブに利用されている．加えて正常終了したジョブ数は全体の 57%を占めるものの，CPU 時間は全体の 1%以下である．

3.2 ジョブの特徴の違い

3.1 節で述べたエラージョブまたはキルジョブと，正常終了ジョブの持続時間，タスク保持数の比較をそれぞれ図 3.2.1 と図 3.2.2 に示す．

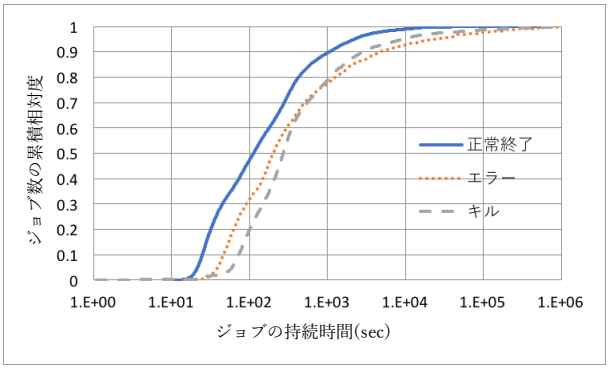

図 3.2.1:主要な終了パターン別ジョブの持続時間分布

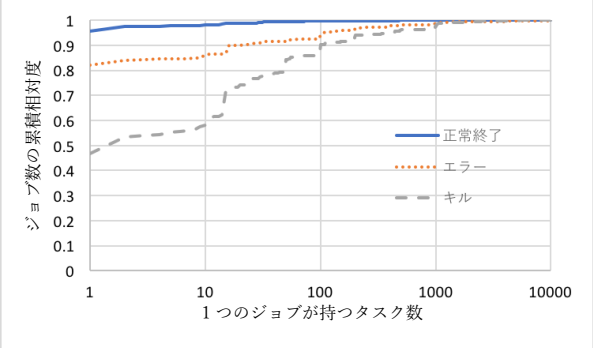

図 3.2.2: 主要な終了パターン別ジョブのタスク保持数分布

図 3.2.1 は図 3.1.1 と同じ手順で持続時間の累積分布を，終了種類ごとに分けて作成した（実行中ジョブは省略）．3 種類のジョブは総数が異なるため，それぞれの累積相対度数をプロットしている．この図から正常終了ジョブよりも，エラージョブまたはキルジョブは持続時間が長いものが多く含まれる割合が高いことがわかる．図 3.2.2 は図 3.1.2 と同じ手順でタスク保持数の累積相対度数を，終了種類別にプロットした．正常終了ジョブよりも，エラージョブまたはキルジョブは多くのタスクを持つものが多く含まれる割合が高いことが分かる．図 3.1.1 と図 3.1.2 で示したように，正常終了したジョブは規模が小さく実行時間が短いため，表 3.1.2 で示したように CPU 時間が少なくなったと考えられる．

4. 異常終了タスクの検知

4.1 スケジューリングイベントの回数

タスクに対しても，ジョブと同様にイベント列の解析を行う．単一のタスクが発生させるイベント数の累積分布を図 4.1.1 に示す．

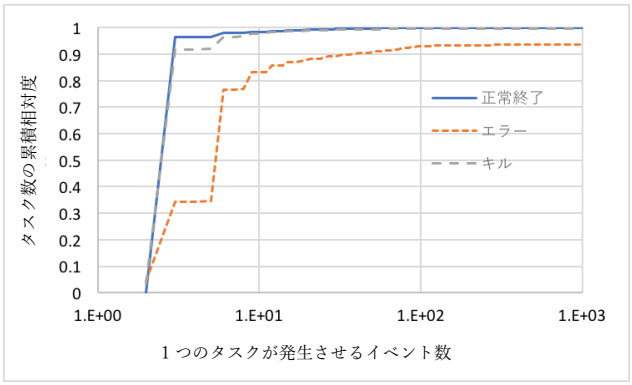

図 4.1.1: 成否別のタスクのイベント数の分布

正常タスクは大部分がイベント列の長さ 3 以下であるが，エラータスクまたはキルタスクの 20%以上が，長さ 3 を超えるイベント列となる．それらのほとんどは，DEAD 状態から Submit イベントが発生して，PENDING 状態に遷移している．すなわち，ジョブが異常終了した後，再投入されていると考えられる．前節で述べた通り，ジョブが記録するイベント列は 99%以上が長さ 3 以下であり，ジョブが再投入されることは稀である一方，タスクのイベント列は長さ 3 より大きくなることがあり，それは異常終了タスクに多いことが分かる．

4.2 イベント回数に基づく異常タスクの検知

4.1 節の結果を踏まえて，タスクが発生させるイベントの数を基にして，異常終了タスクを検知することを考える．T を事前に定めるパラメータとし，タスクが発生させるイベント数が T を超えた時点で，タスクは異常終了であると判定する．異常終了と判定されたタスクの，異常と判定された時刻から，タスクが実際に異常終了するまでの時刻までの CPU 時間を，（浪費を）削減できる CPU 時間と呼ぶ．図 4.2.1 に T と削減できる CPU 時間（の相対値）を $T = 10,20,...,400$ について示す．

図 4.2.2 は，正常終了のジョブが，誤って異常終了ジョブと判定されるジョブの数を示している．正常終了のジョブのイベント列の長さの最大値は 280 である．したがって，T を 280 以上に設定すると，このデータの場合は正常終了のジョブが誤って異常終了ジョブと判定されることは

なくなる．一方，T をそのような大きな値に設定すると，削減できる CPU 時間は小さくなる．

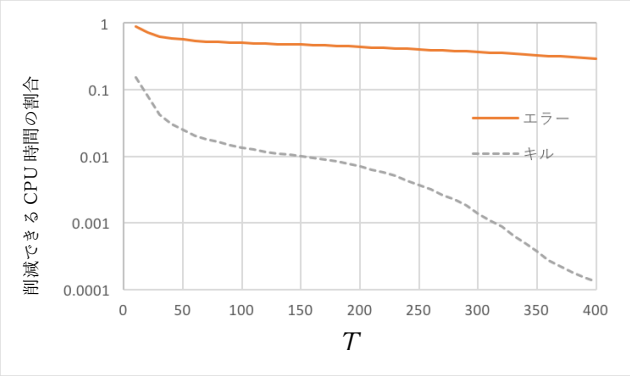

図 4.2.1: 閾値 T と削減できる CPU 時間

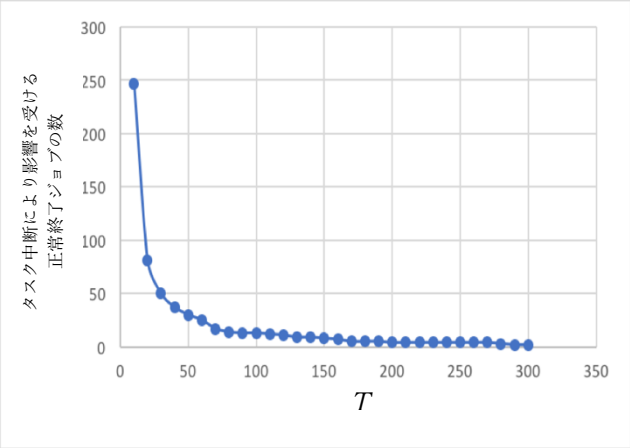

図 4.2.2: タスク中断で影響を受ける正常終了ジョブ数

T=10 のとき，異常終了ジョブによる CPU 時間をエラージョブなら 89%，キルジョブなら 15%削減できる．一方で図 4.2.2 から，246 個 (正常終了ジョブの 0.06%) の正常終了ジョブに属するタスクを中断することになるため，正常なジョブ実行を妨げるリスクがある．T=300 のとき，正常終了ジョブに含まれるタスクを中断させることはない．しかし，削減できる CPU 時間はエラージョブなら 36%，キルジョブなら 0.1%にとどまる．

6. まとめ

本稿では，Google が公開したクラウドプラットフォームのトレースデータである Google Cluster-Usage Traces を解析した．解析から，まず投入されたジョブの 40%が正常終了しておらず，利用された CPU 時間のうち 44%が正常終了していないジョブによって占有されていたことが判明した．次に正常終了したジョブとそうでないジョブについて，ジョブを構成するタスクを比較すると，イベント数に特徴的な違いがあり，正常終了できないジョブには，繰り返し再投入されるタスクが多く含まれることが判明した．

タスクが発生させるイベントの数を基にして，異常終了タスクを検知する手法について検討した．イベント数の閾値を 10 とすると，エラージョブによる CPU 時間を 89%，キルジョブによる CPU の占有を 15%削減できるが，正常終了ジョブも誤って中断させることになる．一方でイベント数の閾値が 300 のときは，正常終了するジョブを中断させるリスクはなくなるが，削減できる CPU は 1%にとどまると判明した．

本研究では，ジョブやタスクのイベント数の観点から，異常終了ジョブについて検討を行った．Google Cluster-Usage Traces には他にも様々な項目があり，それらも考慮した異常終了ジョブの検知手法の提案が今後の課題である．

参考文献

[1] http://googlecloudplatform-japan.blogspot.jp/2016/11/tokyo-region-now-open.html
[2] https://cloud.google.com/storage/docs/
[3] C. Reiss, A. Tumanov, R. Ganger, H. Katz, A. Kozuch, Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In Proceedings of the Third ACM Symposium on Cloud Computing, p.7, 2012.
[4] Z. Liu, S. Cho, Characterizing Machines and Workloads on a Google Cluster. In Eighth International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, pp.397-403, 2012.
[5] S. Di, D. Kondo, W. Cirne, Characterization and Comparison of Google Cloud Load versus Grids. In IEEE International Conference on Cluster Computing, pp.230-238, 2012.
[6] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, Dynamic Energy-aware Capacity Provisioning for Cloud Computing Environments. In Proceedings of the 9th international conference on Autonomic computing 2012, pp.145-154, 2012.