4L-08

クラウドソーシング実行計画の変更時におけるタスク結果キャッシュの利用

鈴木 裕亮 †　　松原 正樹 ‡　　森嶋 厚行 ‡

† 筑波大学 情報学群 情報メディア創成学類　　‡ 筑波大学 図書館情報メディア系

1 はじめに

クラウドソーシングを用いた大量のデータ処理は，機械による処理と異なり，実行が終了するまでに何週間もかかることがある．したがって，様々な理由により，途中で再実行をしたくなる事がしばしばある．特に，複数のタスクを組み合わせたクラウドソーシング応用プログラムにおいては，プログラムがクラッシュした場合や，途中でロジックを変更したくなった場合，途中結果を見て，スパムに気づいた場合などがある．しかし，クラウドソーシングは人手を必要とするため，一から再実行を行なうと作業に時間的・金銭的に高いコストが余計にかかる．

本論文では，複数のタスクを組み合わせたクラウドソーシング応用プログラムを対象に，タスク結果のキャッシュを利用して，低コストで再実行をする手法を提案する．これまでも，クラウドソーシング応用プログラムがクラッシュした場合等にタスク結果キャッシュを利用する手法がすでに提案されている [1][3]．しかし，タスクの変更が行なわれた場合には，タスク結果キャッシュをそのまま利用することができない．

本論文では，複数のタスクを用いたクラウドソーシング応用プログラムを表す実行計画のモデルを定義し，プログラムの実行途中でこの実行計画が変更された場合に，効果的にタスク結果のキャッシュを利用する手法について検討を行なう (図 1)．さらに，どのタイミングで実行計画を切り替えれば良いかに関する分析をおこなった結果を示す．

Avnur ら [2] は，データベースの実行計画において，結合演算処理の切り替えを動的に行なう研究を行なっているが，対象は結合演算処理に限定されている．我々の知る限り，実行計画の変更に伴う再実行時にタスク結果のキャッシュを活用する研究は存在しない．

2 実行計画とキャッシュのモデル化

本論文では，直列のワークフローに限定して考える．実行計画は，図 2 左のような有向グラフで表す．タスクをエッジで表し，各タスクの入出力データをノードであらわす．形式的には，データ (ノード)$D = \{d_1, d_2, ..., d_n\}$

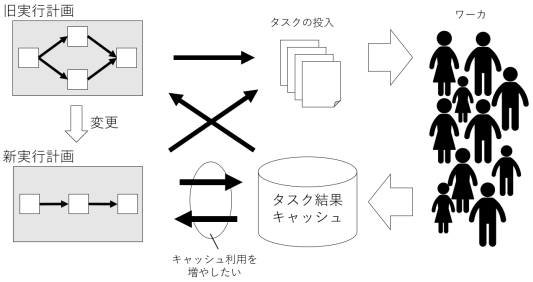

図 1: 実行計画変更時のキャッシュ利用．実行計画はデータをノードで表し，タスクをエッジで表す．

と，タスク (エッジ)$T = \{t_1 = (d_1, d_2), t_2 = (d_2, d_3), ..., t_{n-1} = (d_{n-1}, d_n)\}$ で表現する．各タスクを一つ委託するために必要なコストを $C = \{c_1, c_2, ..., c_{n-1}\}$ とする．

以下では，変更前の実行計画と変更後の実行計画を扱うが，変更後の実行計画はアポストロフィをつけて d'_1, t'_1, c'_1, のように表現し区別する．図 2 では，点線が変更後の実行計画を表す．

タスク結果キャッシュ. 実行計画において，各 d_i $(i > 1)$ はタスク (d_{i-1}, d_i) のタスク結果に相当する．各 d_i に関して，タスク結果キャッシュに格納されている個数を p_i と表現する．

3 キャッシュの効率的な利用

新しい実行計画におけるタスク結果キャッシュの活用においてもっとも望ましいのは，これまで行なった結果 (タスク結果キャッシュ) が，新しい実行計画でできるだけ多く活用されることである．そのための指標として，**キャッシュ貢献率**を次のように定義する．

$$Cache\_Contribution = \frac{1}{n-1} \times \sum_{i=1}^{n-1} \frac{P'}{p_i}$$

ここで，P' は，「旧実行計画の結果から求められた，新実行計画におけるタスクの結果」の数である．キャッシュ貢献率は，旧実行計画のタスク結果 (キャッシュ) のうち，どれぐらいの割合が，新実行計画のタスク結果の計算に貢献できたかを表す．

特に工夫をしなければ，全く同じタスクで無ければ，キャッシュ貢献率は 0 になる．したがって，これを上げるためのルールを用意する (図 2)．

【ルール 1】複数のタスクを一つのタスクに変更する場合，複数のタスク結果キャッシュを合成した結果を

A Task-Result Cache Method in Crowdsourcing Execution Plans
†Yusuke Suzuki, University of Tsukuba
‡Masaki Matsubara, University of Tsukuba
‡Atsuyuki Morishima, University of Tsukuba

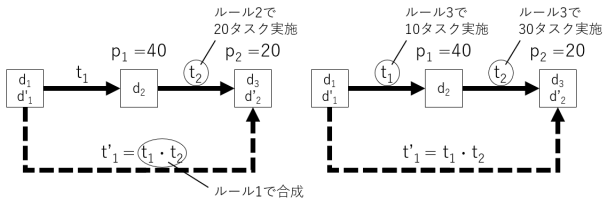

図 2: ルール適用例．データをノードで表し，タスクをエッジで表す．左はルール 1,2 の適用例，右はルール 3 の適用例である．

変更後の実行計画のタスク結果キャッシュとして利用する．

【ルール 2】旧実行計画中で未処理のデータがある場合，ルール 1 の適用前に，特定データノード (図では d_2) 以降の未処理のデータに関してタスクを行なう．

【ルール 3】実行計画の切り替え前後の対応するタスクで入出力の粒度 (個数) が違う場合，ルール 1 の適用前に，キャッシュのデータ数が切り替え後の粒度で割り切れる数になるようにタスクの処理を行なう．

4 キャッシュ貢献率とコストのトレードオフ

前節のルールを適用すると追加のタスクが必要になることがある．したがって，キャッシュ貢献率の向上とコストはトレードオフとなる．特に，ルール 2 においては，どのデータノード以降のデータに対するタスクを行なうかの選択肢が存在する．ここでは，新実行計画にタスクが一つしかないと仮定し*，$d_{n-(i-1)}$ 以降のタスクを全て実行する選択肢のことを切替計画 i と呼ぶ．切替計画 1 は追加タスクを行なわない．

定理 1. 特定のデータノード以降のデータに対するタスクを全て行なうか行なわないか以外の切替計画が，それらの切替計画よりコストを低くすることは無い．□

定理 2. $p_0 = N$ (初期データ数) の時，切替計画 k を利用した際の総コスト (旧実行計画で既にかかったコスト，追加タスクコスト，新実行計画でのコスト) は，$\sum_{i=1}^{n} c_i \times p_i + \sum_{i=n-k}^{n-1} c_i \times (p_{n-k} - p_i) + c'_1 \times (N - p_{n-k})$ である．□

定理 3. 最低の総コストをもたらす切替計画は，旧実行計画のタスクのコスト分布から決定可能である．□

シミュレーション. データノードの数 n が 100 のときを想定し，コスト分布が異なれば，実行計画変更の総コストを最も低くする切り替え計画も異なることをシミュレーションで示す．

本シミュレーションで用いた旧実行計画における各タスクのコスト分布は次の 5 つである．コスト分布 1: $c_i = 0.0\dot{1}$，コスト分布 2: $c_i = i \times 0.00\dot{0}\dot{2}$，コスト分布 3: $c_i = i \times (-0.00\dot{0}\dot{2}) + 0.0\dot{2}$, コスト分布 4: $c_i = 0.035(1 \leq i \leq 20), c_i = 0.00\dot{0}379746835443(21 \leq i \leq 99)$, コスト分布 5: $c_i = 0.00038(1 \leq i \leq 79), c_i = 0.035(80 \leq i \leq 99)$.

*この仮定は議論の一般性を損なわない．

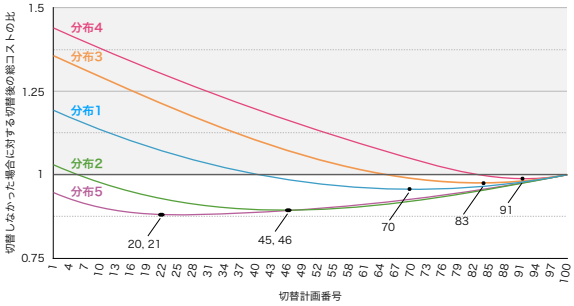

図 3: タスクコストの分布での比較．灰色 (縦軸の値が 1 以上) の領域は旧実行計画ですべてのタスクを行なった方がコストが低いことを表す．黒丸の数値は各分布においてコストが最小となる切替計画番号を表す．

旧実行計画でのタスクのコストの合計は 1.0，新実行計画でのタスクのコストの合計は 0.7 とした．

コスト分布毎のグラフを表したものが図 3 である．横軸は各切替計画で，縦軸は切り替えしなかった場合 (切替計画 100) に対する切り替え後の総コストの比である．黒丸の数値は各分布においてコストが最小となる場合の切替計画番号を示している．それぞれのコスト分布の場合で最もコストが低くなる切替計画が違い，それぞれの切替計画が有用であることが示されている．

5 まとめと今後の課題

本論文では，クラウドソーシング実行計画を途中で変更する際に，タスク結果キャッシュを用いてコスト削減させる問題に取り組んだ．また，キャッシュ貢献率向上とコストのトレードオフについて議論を行なった．今後の課題として直列でない実行計画のコストの最適化について検討することを予定している．

謝辞. 本研究の一部は JST CREST (JPMJCR16E3) による．

参考文献

[1] Little, Greg, et al. “Turkit: human computation algorithms on mechanical turk.” Proceedings of the 23nd annual ACM symposium on User interface software and technology, ACM, 57–66, 2010.

[2] Avnur, Ron, and Joseph M. Hellerstein. “Eddies: Continuously adaptive query processing.” ACM sigmod record, 29, 2 : 261–272, 2000.

[3] Marcus, Adam, et al. “Platform considerations in human computation.” Workshop on crowdsourcing and human computation, 1–4, 2011.