6ZB-05

スマホでグリッドプロジェクト
―みんなのスマホで大規模計算―

小谷 晏経† 島田 雄気† 柳川 陸† 植月 修‡ 福永 圭佑‡
名古屋 清次‡ 佐原 潤哉‡ 瀬之口 潤輔† 石畑 宏明†
東京工科大学 コンピュータサイエンス学部† 伊藤忠テクノソリューションズ株式会社‡

1. はじめに

近年，AI やシミュレーション解析に関する技術の発達に伴い，その計算規模も増大している．一方，企業は数万台のスマートフォンを IT 資産として保有している場合があるが，その利用効率は夜間や休日等に低下する．スマートフォンの性能も年々向上していることから，これらを計算資源として大規模計算の並列処理に活用することにより，IT 資産の利用効率の向上と大規模計算の高速化の一石二鳥を期待する．我々は，大規模なプログラムを複数のデバイスで並列処理するシステムを開発した．さらに，本システムに株価予測シミュレータを適用し，実験と評価を行った．

2. システムの構造

本システムでは，まず事前にオリジナルの大規模計算プログラムを複数の小処理（ジョブ）に分割し，ジョブをノードとする DAG（有向非巡回グラフ）で表現する．次にサーバが待機中のデバイスへ実行可能なジョブを随時送り，返ってきた処理結果を統合して DAG の進捗を管理し，処理を進めていく．

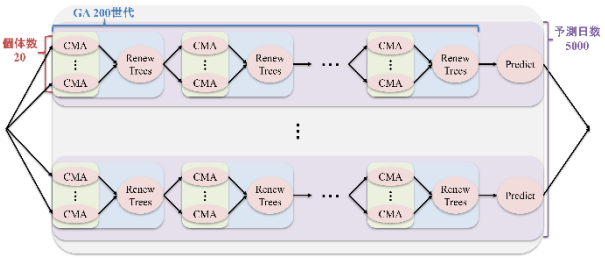

図 1: 株価予測シミュレータの DAG 構造

図 1 は，大規模計算のサンプルとして用いる独自開発の株価予測シミュレータ[1]を DAG で表現したものである．シミュレータは決定木の学習と翌日の TOPIX 先物価格の対数変化率の予測を約 5000 日分行うバックテストプログラムである．予測では決定木を用いて過去数千件のデータを分類し，当日と同じクラスのデータから回帰予測する．決定木の学習には Bi-Level GA[1]を用いる．まず決定木の分岐用の変数をランダムにいくつか選択し，各分岐における閾値を CMA-ES（共分散行列適応進化戦略）により最適化する．これを 20 個体用意し，GA（遺伝的アルゴリズム）により 200 世代かけて予測が良い個体を選別していく．シミュレータはウォークフォワードモデルであり，各日の処理は独立しているほか，各 CMA も独立しているため，いずれも並列処理が可能である．結局，図 1 の楕円で示した CMA，木の更新(RenewTrees)，予測(Predict)の 3 種のジョブからなる DAG として表現される．

2.1. DAG の管理

まず本シミュレータを，cma()や renew_trees()といった関数を組み合わせた Python プログラムとして定義し直す．この作業はセキュリティ・技術的な観点から，自動ではなく手作業で行うことが現実的と考える．この関数がジョブのベースであり，同じ関数でもジョブにより入出力が異なる．例えば CMA ジョブの入力は決定木に用いる変数と予測日，出力は最適化された閾値等である．次に先頭（入次数が 0）のジョブの入出力を手作業で用意する．今回はランダムに生成した木の変数組み合わせと予測日である．以降はジョブの出力が接続先のジョブの入力となる．入力が用意できたジョブを待機状態とすることで，待機中の全てのジョブが並列処理可能となる．

また，ジョブの優先順位を幅優先で事前に決定することで，並列処理の効率を向上させる．DAG の幅優先ソート手法として Kahn's Algorithm が知られているが，今回はそれをノードの次数からソートする代わりに決まった形式のプログラムから DAG 構造を解析・処理できるように実装した．

2.2. デバイス上での処理

各デバイスは Pyodide を用いた Web アプリケーションを介し，ジョブを Python 関数としてブラウザ上で実行する．Pyodide はブラウザ上で Python プログラムを実行できるライブラリである．WebAssembly を用いており，OS に依存せずネイティブ相当の処理が可能となる．

アプリではまず初期化処理として，サーバから関数と参照するデータの取得，Pyodide の初期化等を行う．その後サーバから関数の入力パラメータを取得し，計算処理を行い，結果をサーバに送信する．すべてのジョブが終わるまで，パラメータ取得以降の工程を繰り返す．

3. 評価実験

株価予測シミュレータに対し，いくつかの条件下で処理時間を計測する．分散処理用のスマートフォンとして iPad Air 4 と Zenfone 8 各 3 台の合計 6 台を使用する．比較用に IaaS インスタンスとして AWS EC2 m4.large を用いて分散せずオリジナルプログラムを実

Project for Grid Computing with Smartphones – Large-scale computing with your smartphones –
Yasunori Kotani † Yuki Shimada † Riku Yanagawa † Osamu Uetsuki ‡ Keisuke Fukunaga ‡ Seiji Nagoya ‡ Junya Sawara ‡ Junsuke Senoguchi † Hiroaki Ishihata †
† School of Computer Science, Tokyo University of Technology
‡ ITOCHU Techno-Solutions Corporation

行した場合も検討する．なお，m4.large の性能は概ね最新の個人向けハイエンド CPU の 1/3 程度である．

3.1. 処理効率の変化

予測日数，GA 世代数，CMA 世代数，及び学習件数を変えた 6 つのケースを用意する．DAG 構造より，予測日数は並列の，GA 世代数は直列のジョブ数にそれぞれ線形に関与する．これらを増やすことで処理時間も線形に増加することが予想される．なお全ケースで木の個体数は 20 に固定しており，最大の並列ジョブ(CMA)数がデバイス数を上回る．よってほぼ常時全デバイスが活用され，並列・直列による差異は生じないはずである．CMA 世代数は CMA 進化計算の世代数であり，CMA ジョブ内のループ回数である．全体の計算量は変わらないがジョブ数も変わらないため，ジョブの管理等に要するオーバーヘッドを無視できる．学習件数は回帰式の学習に用いる過去データの件数であり，配列の大きさである．これも計算量は変わらないが，NumPy 等の Python ライブラリでは C 言語実装やマルチスレッド等により配列処理が効率化されているため，Pyodide でもその効率化が有効であるか確認する必要がある．

各ケースについてパラメータの組み合わせと，それらに対する m4.large 非並列と 6 台並列による処理時間を表 1 に示す．ケース A は基準ケースである．B-D はそれぞれ計算量を A の 5 倍になるようパラメータを 1 つずつ変化させたもの，E は学習件数を減らしたもの，F は 2 つの世代数を同時に大きくしたものである．

表 1: オリジナルとの処理時間の比較

Case	予測	GA	CMA	学習	m4(s)	6 台(s)
A	2	2	2	2500	**132**	**119**
B	**10**	2	2	2500	**663**	**214**
C	2	**10**	2	2500	**661**	**232**
D	2	2	**10**	2500	**632**	**185**
E	2	2	2	**100**	**31**	**30**
F	2	**50**	**10**	2500	**23688**	**4816**

BC 間の処理時間はほぼ等しく，ジョブの直列並列による差異は無いと言える．ところが A と比較すると，m4 の処理時間はちょうど 5 倍である一方，6 台並列時は 2 倍程度に抑えられている．F では更に効率化されている．これはジョブが多い程初期化処理等によるオーバーヘッドの影響が軽減されるためと考えられる．D では更に高速化が認められることから，通信時間やシステム側のジョブ管理に関するオーバーヘッドが多少あると思われる．これらのオーバーヘッドについて次節で詳しく考察する．A と E の比較では学習件数が増えても高速化は維持されていることから，NumPy 等の効率化の恩恵を Pyodide でも受けられていると考えられる．実際，処理中のデバイスの CPU 使用率を確認したところ，複数スレッドが活用されていた．

3.2. デバイス数による処理効率の変化

まず全体の処理時間を定式化する．ワーカー側の初期化処理では数十秒のオーバーヘッドがあることを考慮し，初期化処理時間をt_{init}，ノード数をN_{node}，デバイス数をN_{device}とし，1 つのジョブの平均処理時間をt_{job}とすると，全体の処理時間t_{all}は次式で推定できる．

$$t_{all} \approx t_{init} + \frac{N_{node}}{N_{device}} t_{job}. \quad (1)$$

次に個体数を 5 に減らしたケース A’を追加し，B と合わせて台数を変えて計測す（表 2）．ただし iPad と Zenfone の性能等の差を減らすため，これら 2 台を 1 セットとし，1-3 セットで比較する．

表 2: 台数による処理時間の比較

Case	予測	個体	2 台(s)	4 台(s)	6 台(s)
A’	2	5	**47**	**35**	**31**
B	10	20	**604**	**342**	**214**

ケース A’の 3 つの計測結果を式(1)に当てはめると，初期化処理時間は誤差なく$t_{init} = 23$秒となる．実際，これを処理時間から減算すると，処理時間と台数はちょうど反比例する．初期化処理時間が 23 秒程度であるため，ケース B のように数百秒かかる処理では殆ど影響を受けず，反比例に近い結果となっている．

また，ジョブの中身を空にして 2 台で計測したところ 174 秒かかった．単純にジョブ数 430 で割れば，t_{init}以外におよそ 0.4 秒/ジョブのオーバーヘッドがあることになる．$t_{job} \approx 1$秒であるから，この割合は大きい．t_{job}が数十秒以上かかるタスクが適していると言える．

さらに，ケース B について表 1 も合わせて考察すると，2 台並列でも m4.large より高速であり，スマートフォンには実用的なレベルで計算資源としての能力と経済的価値があると言える．なお，本来の株価予測モデルは予測日数約 5000 日，GA 200 世代，CMA 500 世代と非常に大きいパラメータを使う．計算量としてはケース F の 50 万倍である．ここまでの実験結果から推定すると，m4.large では約 375 年かかるが，1 万台のデバイスで並列処理を行うことで 17 日程度の現実的な時間に収まる計算になる．

以上の結果から，本システムが得意とするタスクの特性として，最大の並列ジョブ数が十分多いこと，繰り返しはジョブ数を増やさずジョブ内で増やすこと，NumPy 等の配列処理を活用すること等が挙げられる．そのようなタスクとしては，今回の GA のような進化計算の他に，ハイパーパラメータのグリッドサーチ等が挙げられる．

4. おわりに

大規模な計算をスマートフォンで並列処理するシステムを開発した．既存の IaaS と処理時間を比較することで，本システムが得意とする処理の特性を検討し，スマートフォンの計算資源としての能力と経済価値が十分あることを確認した．現在，システムの品質向上と共にクラウドサービス等の利用を含め，安価な高性能コンピューティングシステムとして商用化を検討している．

参考文献

[1] 瀬之口潤輔，“多進木 Bi-Level GA によるノイズを含む空間の抽出と複雑系データの予測，” 人工知能学会論文誌 36.5, pp.F-L52_1-7, 2021.