小口研究室　研究紹介　(2019年度)
(お茶の水女子大学理学部情報科学科)

エッジ，クラウド間分散処理に向けた動作識別手法の検討
(研究担当：高崎　智香子)

研究背景

防犯カメラなどの動画像データの活用
ディープラーニング技術の応用
- 正確な識別処理には大量のデータ収集・処理が必要
- リアルタイムに識別処理を行うのは非常に困難

➡ センサとクラウドで識別処理を分散

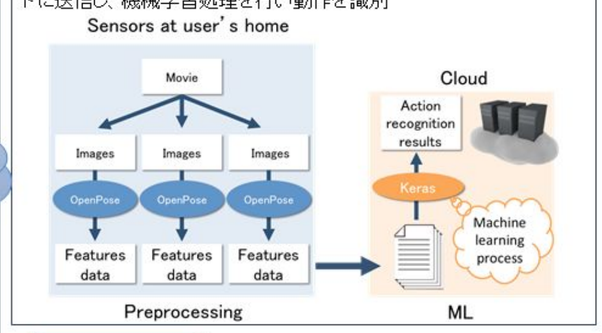
センサ側における前処理、
クラウド側における識別処理の並列化
によって負荷分散

OpenPose
- 姿勢推定ライブラリ
- 135の特徴点を認識可能
- 特殊センサを使わずに解析

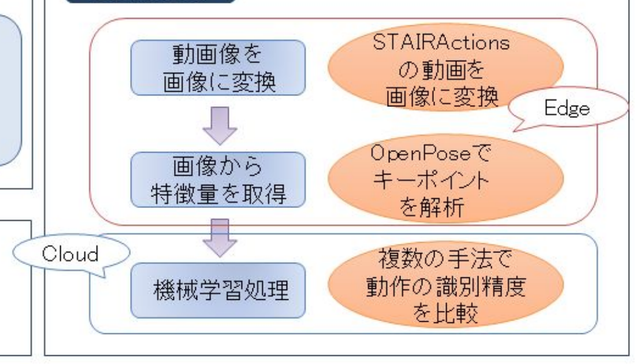
Keras
- NN実装ライブラリ
- GPUで高速に動作
- モデルの記述が容易

研究概要

各家庭のセンサで収集した動画像データを静止画に変換後、OpenPoseを用いて特徴量データに変換。特徴量データのみをクラウドに送信し、機械学習処理を行い動作を識別



実験概要

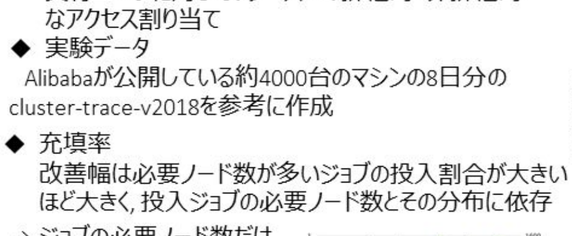


実験結果

各データセットを用いて100カテゴリの動作識別精度を測定

モデル	画像数	間隔	データ数
(1a) NN	10	0.1 sec	87923
(1b) NN	10	0.3 sec	96807
(2a) LSTM w/10 steps	10	0.1 sec	87923
(2b) LSTM w/10 steps	10	0.3 sec	96807
(2c) LSTM w/20 steps	20	0.1 sec	128039
(2d) LSTM w/30 steps	30	0.1 sec	85553

- NNとLSTMによる識別精度の比較



Top 5 Accuracyで測定
(予測した上位5カテゴリに正解が含まれる精度)

考察

1. 識別精度の向上
- データセット2cのデータ数が最多 → 精度が最も高い
- 動作カテゴリを3から100に増加させることが、過学習の抑制に効果的 →人間の動作を汎用的に学習できた
- 損失の収束が学習時でも2～2.5程度
  → キーポイントのみでの動作識別の難しさ

2. 学習処理時間の調査
- センサ側の処理（OpenPoseによる特徴量抽出）
  画像1枚あたり平均0.089秒 → 1秒間に約11枚処理可能
- クラウドでの処理（学習済みモデルによる推論）
  1000データあたり0.03～0.20秒

→センサ側での処理が重い

3. 3D ResNetとの比較による提案システムの評価
- 3D ResNetとは

CNN を非常に深く重ねることを可能にしたResNetモデルに時間軸を追加することで動画に応用
- 1動画あたりの平均処理時間：7.93 秒
- 識別精度：0.849

→ 3D ResNetは短時間でより高精度な識別が可能

➢ 提案システムの利点：ポーズデータの再利用可能性
- 人物の動作の癖を学習し、家族を特定
- 異常な動作を未然に検知

まとめと今後の課題

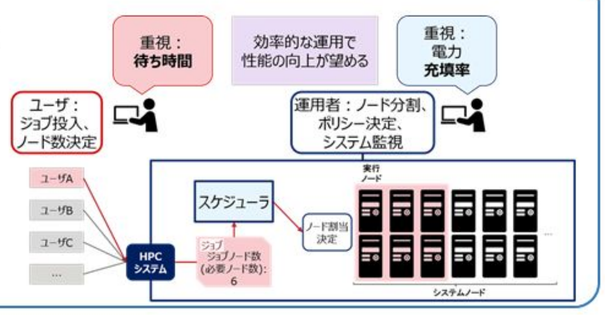
まとめ
- 動画をOpenPoseで前処理後、機械学習で100の動作識別
- OpenPoseによる前処理がボトルネック
- 提案システムで利用する関節点データの再利用可能性

今後の課題
- ボトルネックであるOpenPoseの処理時間の調査、高速化
- 家庭で利用されるエッジデバイスを用いたより実環境に近い環境での評価

高充填率と低待時間を両立するジョブバッチスケジューリング手法の提案
(研究担当：高山　沙也加)

研究背景

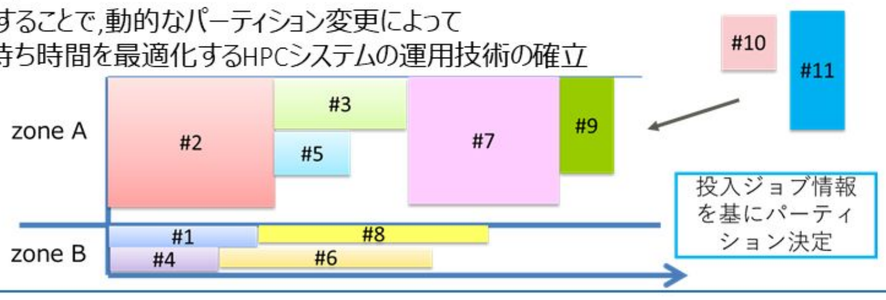
- ◆ 近年のHPCシステムでは投入されるジョブ数は増加しており，要求条件も多様化
  - ◆ システム規模は増加傾向
- ◆ 運用者とユーザでは重視する点は異なる
  - ◆ 運用者：電力，充填率
  - ◆ ユーザ：待ち時間
- ◆ 適用するスケジューリングアルゴリズムによって充填率や待ち時間は大きく変わる



研究目的

- ◆ どの程度の必要ノード数のジョブであれば，ある規模のシステムで充填率と待ち時間の悪化なしに実行できるか？
- ◆ システムの大きさがどれくらいであれば充填率と待ち時間の悪化なしにある大きさのジョブを実行できるか？

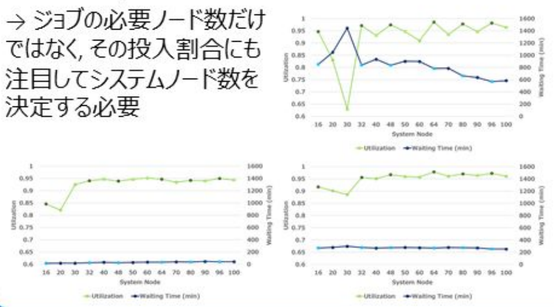
を明らかにすることで，動的なパーティション変更によって
充填率や待ち時間を最適化するHPCシステムの運用技術の確立



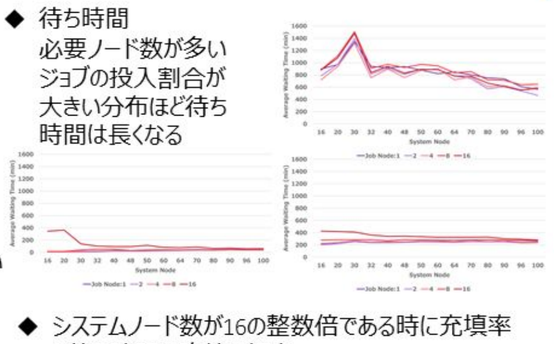
実験・分析結果

- ◆ Slurm Workload Manager
  オープンソースのジョブスケジューラ
  実行ユーザに対してリソースへの排他的・非排他的なアクセス割り当て
- ◆ 実験データ
  Alibabaが公開している約4000台のマシンの8日分のcluster-trace-v2018を参考に作成
- ◆ 充填率
  改善幅は必要ノード数が多いジョブの投入割合が大きいほど大きく，投入ジョブの必要ノード数とその分布に依存

→ ジョブの必要ノード数だけではなく，その投入割合にも注目してシステムノード数を決定する必要



- ◆ 待ち時間
  必要ノード数が多いジョブの投入割合が大きい分布ほど待ち時間は長くなる



- ◆ システムノード数が16の整数倍である時に充填率と待ち時間の改善の傾向
- ◆ システムノード数とジョブの必要ノード数の比が小さくなることで必要ノード数毎のジョブの投入しやすさの違いも小さくなる

	Alibaba	反転	均等
全システムノード数の平均充填率	0.927	0.924	0.950
全システムノード数の平均待ち時間 (min)	31.61	840.29	272.27

今後の課題

- ◆ 投入ジョブの必要ノード数とそのジョブ分布が充填率と待ち時間に大きく影響
  →投入ジョブの必要ノード数やジョブ分布を参考にしたパーティション分割アルゴリズムの考案
- ◆ システムノード数が必要ノード数の公倍数の整数倍の時スケジューリング改善
  →必要ノード数と分布から複数ジョブを1つの塊とするグルーピング手法

完全準同型暗号を用いたFP-growthによる頻出パターンマイニングの実装の改善
(研究担当：種村　真由子）

研究背景

- **ビッグデータの利活用**
  IoT分野をはじめ、各種ビジネスなどにおいて大規模データの収集・活用が進んでいる
- **大規模データ処理の外部委託**
  膨大なデータ処理には高性能な計算機が必要であり、外部（クラウド等）に計算を委託するのが現実的
- **情報のセキュリティ管理の必要性**
  信用できない外部サーバにプライバシに関わるデータを平文で置くことは危険である

→外部に平文データを公開せずに委託処理を行う

研究課題・システム概要

完全準同型暗号（FHE）を用いた頻出パターンマイニングのFP-growthによる実装の改善
クライアントと委託先サーバを想定した、頻出パターンマイニングの委託処理システムを作成する。
委託先サーバはFHE暗号化されたデータを復号せず処理し、クライアントに結果を返送する。

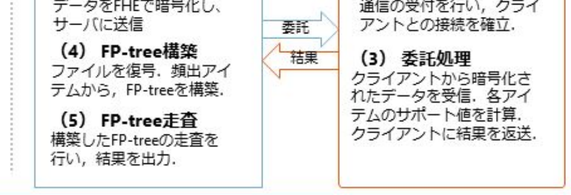
本研究では、FP-growthアルゴリズムを使用したプログラムの実装を改善する

●完全準同型暗号
暗号文同士で加算が成立する**加法準同型性**と、
乗算が成立する**乗法準同型性**をもつ公開鍵暗号方式である。
暗号化したままの比較演算が困難なことと、計算毎に増加する暗号文のノイズが閾値を超えると復号できなくなること、処理の計算量が膨大であることが課題
- Leveled FHE: 指定した回数分の演算について、ノイズが正しく評価できる実装のFHE

●頻出パターンマイニング
トランザクションの集合から、一定以上の頻度で出現するパターンを抽出する手法。代表的なアルゴリズムとしてApriori、FP-growthがある。本研究ではFP-growthを用いる。

	Apriori (関連研究)	FP-growth (本研究)
概要	各パターンの頻出・非頻出をそれぞれ調べる.	頻出パターンを格納した木構造を走査する.
実装	易	難
DBのスキャン回数	頻出パターンに依存	2回（最低の場合）
アクセスのはやさ	遅	速
高コストになるのは	頻出パターンが長い場合	アイテムが多い場合

処理の流れ

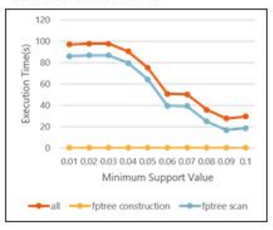


実装・実験

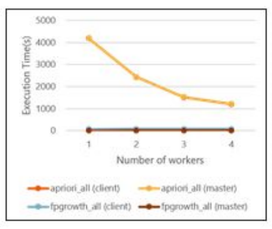
実装・環境
- 主な使用言語：C++
- 使用ライブラリ
  - Boost、Helib（FHEライブラリ）、OpenMPI（MPIライブラリ）

実験
FP-growthの実装改善に向け、現プログラムやライブラリの挙動の傾向を調査する
- 実験データ
  - IBM Quest Synthetic Data Generatorにより作成した人工データ
    アイテム数30、トランザクション数9900
- 実験環境
  - マシン:
    CentOS6.9、Intel® Xeon® プロセッサ E5-2643 v3、3.6GHz、6コア、12スレッド、RAM512GB

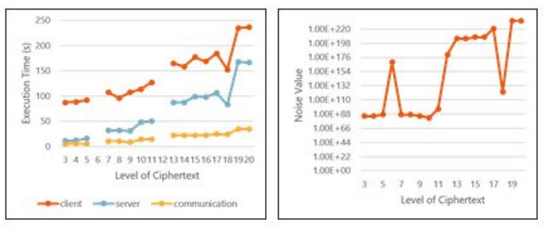
結果1
ミニマムサポート値を変化させた際の実行時間



結果2
Apriorとの実行時間比較



結果3
レベルごとの実行時間、ノイズ



新規検討手法
現システムの実装を変更し、サーバ側の処理を増やす手法を検討している。「処理の流れ」の（3）に「サポート値とミニマムサポート値の差をとる」という処理を追加する。

- **サーバ側の追加処理**
  サポート値の配列<暗号文型>

Sup(0) - minsup	Sup(1)	Sup(2)	Sup(3)	...

暗号文のままミニマムサポートと差をとる

ミニマムサポートとの差の配列<暗号文型> ※新規作成

diff(0)	diff(1)	diff(2)	diff(3)	...

→この後　クライアントに返送
diff(n) ≧ 0 →頻出　／　diff < 0 →頻出でない　とする

今後の課題

- 新規に検討している手法の実装
- FP-growthをFHEにより適した形に改善するなどの方法で、よりサーバ委託部分を増加させる方法の検討

http://ogl.is.ocha.ac.jp/
oguchi@is.ocha.ac.jp