2021年1月20日修正版

Optuna分散学習@「不老」Type II サブシステム 補足（DBの起動に1ノード余計に使わない方法）

概要

- 2020年10月23日に実施した「スーパーコンピュータ「不老」Type IIサブシステム利用Optuna利用講習会」にて、「不老」におけるOptunaを用いた分散機械学習の方法を紹介した
  - 講師による公開資料
  - https://www.slideshare.net/pfi/20201023-optunalectureatnagoyauni-238946529
- 大変便利な使い方ではあるが、上記で紹介した方法ではPostgreSQLサーバを起動するだけのために計算ノードを占有してしまい、計算資源と利用ポイントがもったいないという問題がある
  - cx-shareを使えばマシ（利用ポイント1/4）になるが、それでもかなりもったいない
- そこで、PostgreSQLの起動のためだけに1ノード余計に使わなくても良い方法を紹介する

- さらに補足
  - ログインノードでPostgreSQLを立ち上げるという手段も可能ではあるが、ログインノードは多くのユーザが利用するため、ポートが衝突することがあり、プロセスが残留したりすると迷惑がかかるため、推奨はできない

基本的な考え方

- 1ノードだけPostgreSQLプロセスと機械学習プロセスが共存したノードを用意する
- 最初にPostgreSQLを起動してから機械学習を行うバッチジョブを投入
- その後従来通りに他ノードのPostgreSQLを参照して機械学習を行うバッチジョブを投入

- 補足
  - PostgreSQLを起動しているジョブが終了してしまうと他のジョブにも影響が及ぶが、それは独立したノードでPostgreSQLを起動した場合も同様なので気にしないことにする。
  - PostgreSQLを起動しているノードだけ余計な負荷がかかることになるが、たいした処理をしているわけではない。そこをセンシティブに考える必要がある場合は諦めて別ノードでPostgreSQLを起動すること。

バッチジョブスクリプトの具体的な書き方

従来のPostgreSQLを起動するバッチジョブスクリプト（#PJM行は省略）

```
module load singularity
singularity build postgres.img docker://postgres:9.5.20-alpine

mkdir postgres_data 2>/dev/null
mkdir postgres_run 2>/dev/null
cat $PJM_O_NODEINF > rdb_server_nodeinfo
# PostgreSQLを起動する、PostgreSQLのプロセスが死ぬまでまたされる
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img /docker-entrypoint.sh postgres
pg_ctl -D /var/lib/postgresql/data -l logfile start  ←この行は実際には使われていない
```

円記号は本来はバックスラッシュ。行末にバックスラッシュを置くと本来は1行で書くべきコマンドを折り返して記述することができる。
（以下のスライドでも同様。）

従来のPostgreSQLを参照して機械学習を行うバッチジョブスクリプト（#PJM行は省略）

```
module load singularity
singularity exec --nv ./optuna.sif python pytorch_rdb.py
```

PostgreSQLを起動してから機械学習を始めるバッチジョブスクリプト

基本的には2つのジョブを繋げるだけ

```
module load singularity
singularity build postgres.img docker://postgres:9.5.20-alpine

mkdir postgres_data 2>/dev/null
mkdir postgres_run 2>/dev/null
cat $PJM_O_NODEINF > rdb_server_nodeinfo
# PostgreSQLの初回起動準備だけを行う（既に行ってあれば不要だが、いずれにせよすぐに終わる）
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img /docker-entrypoint.sh
# PostgreSQLを起動する、この方法であれば起動した時点で次の行へ進める
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img pg_ctl -D /var/lib/postgresql/data -l logfile start
# 機械学習
singularity exec --nv ./optuna.sif python pytorch_rdb.py
# PostgreSQLを明示的に終了させたければこのように書くのだが、
# 他のジョブからのDB参照もなくなっていることが保証できないと問題が起きるので注意が必要
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img pg_ctl -D /var/lib/postgresql/data -l logfile stop
```

PostgreSQLを起動してから機械学習を始める：準備

- DBの初期設定をあらかじめ行っておく必要がある模様
  - （もっとスマートな解決方法がありそうな気はするのですが）
  - 初期設定（ログインノード上でもインタラクティブジョブでも良い）

```
module load singularity
# コンテナを入手
singularity build postgres.img docker://postgres:9.5.20-alpine

# DBの初期設定用ディレクトリ（一度作成したらずっと使い回せるはず）
# 不具合がある場合はディレクトリを消して作り直すと良い
mkdir postgres_data 2>/dev/null
mkdir postgres_run 2>/dev/null

# PostgreSQLの初回起動準備
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img /docker-entrypoint.sh postgresql
一通り終わったらCtrl+cで終了する
```

出力例

```
（途中まで省略）
PostgreSQL init process complete; ready for start up.

LOG:  could not create IPv6 socket: Address family not supported by protocol
LOG:  database system was shut down at 2021-01-20 19:14:32 JST
LOG:  MultiXact member wraparound protections are now enabled
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
^CLOG:  received fast shutdown request
LOG:  aborting any active transactions
LOG:  autovacuum launcher shutting down
LOG:  shutting down
LOG:  database system is shut down
```

準備後に実行するバッチジョブスクリプト（サーバの起動＋機械学習）

基本的には2つのジョブを繋げるだけ

```
module load singularity
# DBが起動するサーバのIPを記録
cat $PJM_O_NODEINF > rdb_server_nodeinfo
# PostgreSQLを起動する、この方法であれば起動した時点で次の行へ進める
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img pg_ctl -D /var/lib/postgresql/data -l logfile start
# メインの機械学習処理
singularity exec --nv ./optuna.sif python pytorch_rdb.py
# PostgreSQLを明示的に終了させたければこのように書くのだが、
# 他のジョブからのDB参照もなくなっていることが保証できないと問題が起きるので注意が必要
singularity run -B postgres_data:/var/lib/postgresql/data -B postgres_run:/var/run/postgresql ¥
 postgres.img pg_ctl -D /var/lib/postgresql/data -l logfile stop
```

従来通りに他ノードのPostgreSQLを参照して機械学習を行うバッチジョブスクリプト

- こちらは従来のPostgreSQLを参照して機械学習を行うバッチジョブスクリプトそのものを使えば良い
- PostgreSQLを起動するジョブが先に始まっている必要がある点には注意
  - 従来の方法でも同様ではあるが
- もちろん、バルクジョブとして投入しても問題ない
  - PostgreSQLの起動を含むジョブだけ先に投げて（起動させて）おいて、あとはバルクジョブで一斉実行すれば良い