ワークフローシステム Nextflow の TCS 対応と スーパーコンピュータにおける統一的なワークフロー記述

大島 聡史 [1)], 南里 豪志 [1)], 美添 一樹 [1)]

1) 九州大学 情報基盤研究開発センター

ohshima@cc.kyushu-u.ac.jp

TCS Support for the Nextflow Workflow System and Unified Workflow Description on Supercomputers

Satoshi Ohshima[1)], Takeshi Nanri[1)], Kazuki Yoshizoe[1)]

1) Research Institute for Information Technology, Kyushu University

概要

大学情報基盤センター等の全国共同利用スーパーコンピュータシステムではバッチジョブ実行による計算資源の共有利用が行われており、ジョブの記述方法は採用しているバッチジョブスケジューラにより異なる。単純に単一のプログラムを実行するだけであれば、ジョブスケジューラによる差異は利用する計算資源の指定などシンプルだが環境依存性がある部分であり、統一的な記述の必要性は限定的である。しかし複数のジョブの動作順序や依存関係を制御したい場合、ジョブスケジューラに備わった独自の機能で実現可能なこともあるが、環境依存性が高く移植性は低い。ワークフローシステムは依存関係のあるジョブの制御など高度なジョブ実行を可能とするソフトウェアであるが、我々が運用する玄界で用いているジョブスケジューラ Technical Compute Suite(TCS) を含むスーパーコンピュータで広く使われているスケジューラに対応するメジャーなワークフローシステムはなかった。そこで我々はオープンソースのワークフローシステムである Nextflow に TCS 向けの機能を追加実装した。これにより多くのスーパーコンピュータを 1 つのワークフローシステムで制御可能となった。

1 はじめに

国内の大学情報基盤センター群のスーパーコンピュータを始めとして、大規模な共有計算機システムにおいては多数のジョブを円滑に処理するためにジョブスケジューラが利用されている。一般的にジョブスケジューラの中身としては bash などのシェルスクリプトが利用されており、その気になれば複雑なジョブを記述することもできる。

国内の大学情報基盤センター群に設置された全国共同利用システムで用いられているジョブスケジューラの例を表 1 に示す。このように現在はシステムごとに様々なスケジューラが利用されている。ジョブスケジューラごとに記法は異なり、また同じジョブスケジューラでもリソースグループ/ジョブキューの設定などがシステムごとに異なる。そのため、基本的には同じジョブスクリプトをシステム間で使い回すことは難しい。ただしジョブの設定に関する部分以外は普通のシェルスクリプト（bash スクリプト）であるため、まったく使い回しができないわけではない。また、Xcrypt [1] のように複数のシステムで共通に使えるジョブスクリプト言語の研究事例も存在する。

複数のジョブを連携させて実行させたいという要求も一定数存在する。ジョブスケジューラによっては複数のジョブを順序づけて実行する機能などを有しており、一部のユーザはそれらを活用していると思われる。しかしこれらの機能を活用した高度なジョブ制御は複雑であり、ジョブスケジューラの機能に依存するためシステム間で共通化することは難しい。

一方、スーパーコンピュータにおけるジョブ実行に限らず、計算科学やデータ科学などの分野ではプログラムや操作による連携処理を記述するツールとして様々なワークフローシステムが利用されている。（ワークフローツールやワークロードエンジンなどとも呼ばれている。）近年ではスーパーコンピュータ同士の連携やスーパーコンピュータと外部の装置やストレージとの連携が必要とされる機会も増えていることから、スーパーコンピュータ上におけるワークフローシステムの利用も需要が大きくなる可能性がある。

本稿では、オープンソースのワークフローシステム

表 1　大学情報基盤センター群の全国共同利用システムで用いられているジョブスケジューラの例（Web サイトまたは利用者向けドキュメントに記載があったもの）

大学・施設名 / システム名	ジョブスケジューラ
北海道大学 / Grand Chariot 2	PBS Professional
東北大学 / AOBA-S	NEC Network Queuing System V
JCAHPC / Miyabi	PBS Professional
東京大学 / Wisteria/BDEC-01	Technical Computing Suite
東京科学大学 / TSUBAME 4.0	PBS Professional
名古屋大学 / 不老	Technical Computing Suite
大阪大学 / SQUID	NEC Network Queuing System V
京都大学 / Camphor, Laurel, Cinnamon, Gardenia	Slurm
九州大学 / 玄界	Technical Computing Suite

である Nextflow [2] に機能を追加し、我々が九州大学情報基盤研究開発センターにて運用しているスーパーコンピュータ玄界で採用しているジョブスケジューラ Technical Computing Suite(TCS) に対応させた事例を紹介する。以下、2 章ではワークフローシステムと Nextflow の概要について紹介し、3 章では Nextflow に対する機能追加の内容について述べる。4 章では Nextflow を用いてスーパーコンピュータでジョブを実行する例を紹介し、5 章でまとめる。なお本原稿を執筆している時点（2025 年 9 月）ではすでに Nextflow の開発リポジトリに TCS に対応するためのコードが取り込まれているが、一般向けのリリースバージョンにはまだ反映されていないそのため TCS に対応した Nextflow の入手方法を本稿末尾の付録に記す。[*1]

2　ワークフローシステムと Nextflow

スーパーコンピュータのジョブ実行制御に限らず、ジョブや資源を管理する様々なソフトウェアが活用されている。高性能計算に関する分野では、コンテナ計算環境の利用と管理に用いられている Kubernetes や、ブラウザ上で Python を対話的に利用できる Jupyter(Jupyter Notebook, Jupyter Lab) が広く知られている。

Nextflow は主にバイオインフォマティクスの分野で人気のあるワークフローシステムである。当該研究分野では DNA シーケンサから生成されるデータを複数のソフトウェアで順次処理していくといったワークフロー処理の需要が大きく、それらを記述するツールとして Nextflow などが活用されている。Nextflow はワークフロー記述言語と実行エンジンを提供しており、ワークフローの記述には Java 上で動作する Groovy [3] が用いられている。また Nextflow はオープンソースのワークフローシステムであり、GitHub を用いて開発され Apache 2.0 ライセンスで公開されているため、利用者が必要な機能を追加することも可能である。

Nextflow の特徴のひとつとして、パブリッククラウドやスーパーコンピュータで使われている様々なジョブスケジューラ・実行エンジンに対応していることが挙げられる。Nextflow の中で行われるジョブの実行や管理は Executor クラスにより実現されているが、以下のジョブスケジューラ等に対応した Executor がそれぞれ実装されている：

- AWS Batch
- Azure Batch
- Bridge
- Flux Executor
- Google Cloud Batch
- HTCondor
- HyperQueue
- Kubernetes
- Local
- LSF
- Moab
- NQSII
- OAR
- PBS/Torque
- PBS Pro

[*1] プレリリース版である Version 25.08.0-edge には TCS 対応の記載があるが、最新リリース版である Version 25.04.7 では未対応である。オンラインドキュメントには既に TCS に対応している旨の記述がある。`https://www.nextflow.io/docs/latest/executor.html#tcs`

- SGE
- SLURM

(Local はジョブスケジューラではなく Nextflow が動作している環境でのローカルな実行。スレッドによる並列ジョブ実行をサポート。) すなわち Nextflow は表 1 で示したジョブスケジューラのうち TCS 以外を既にサポートしており、TCS に対応すれば Nextflow を用いて全てのシステムにおけるワークフローの実行が可能になる状態であったことを意味する。

Nextflow ではバッチジョブを用いた処理とそうでない処理を混在して記述することができる。そのため例えば、Web 上で公開されている大規模データのダウンロード、計算ノードを用いた大規模なデータ処理、処理済みデータのアップロードを順に行いたい場合、全ての処理をバッチジョブスクリプトに書くとデータのダウンロード・アップロードもジョブに含まれてしまう。(その時間だけ計算ノードを使わねばならず、計算ノードの利用効率が悪化したり利用料金が増加する。) Nextflow ではデータのダウンロード・アップロードはログインノードで行い、データ処理のみ計算ノードでバッチ処理するようなワークフローを扱うことができる。もちろんこのような処理はシェルスクリプトなどで記述することが可能であるが、ジョブスケジューラの仕様と環境 (利用するノードグループやジョブキューなど) に依存した処理の記述が必要となる。Nextflow を使えばジョブスケジューラの差異を気にせず様々なシステムで同様に使えるワークフロー処理を記述できる点が優れている。

3 Nextflow の TCS 対応

本章では Nextflow の TCS 対応のために行った実装の概要を述べる。

上述の通り、Nextflow では Executor クラスによって各ジョブスケジューラに対応する機能を実装している。既に Slurm や PBS などに対応した Executor の実装が存在していたことから、TCS 向けの Executor (TcsExecutor) の実装はそれらを参考にして実装した。具体的に実装した TcsExecutor の備える主な機能は以下の通りである。これらのほとんどは 10 行程度で記述可能な単純な処理であり、合計でも (コメント等を除くと) 100 行程度の小規模なプログラムである。

- TCS の記法にあわせたジョブ投入コマンド文字列の生成
- TCS のジョブ実行ステータスと Nextflow の定義するステータスとの対応付け
- TCS のジョブ実行状況表示のパースと情報の抽出

なお TCS を導入しているシステム同士でも利用する資源 (リソースグループ/キューの名称や利用する資源量の単位 (コア数、ノード数、GPU 数など) など) の指定方法はそれぞれ異なるため、TcsExecutor 内では細かく処理せず、Nextflow の設定ファイルを用いて指定できるようにした。(コア数やメモリ量の指定に合わせた文字列解析機能が実装されている Executor もあったが、TcsExecutor では実装しなかった。) 設定ファイルの例を Listing 1 に記す。`executor` に `tcs` と設定することで TcsExecutor が利用される。`time` は全ての Executor で利用されているジョブの実行時間に関するオプションであり、TcsExecutor でも同様に利用している。`clusterOptions` は汎用の追加オプションであるが、TcsExecutor ではここにシステム依存の計算資源オプション等を書くことにした。なお `clusterOptions` は pjsub コマンドの引数としてそのまま使われるようにしてあるため、`-L` オプション以外にも自由に様々なオプションを記述可能である。

4 実験

Nextflow のサンプルプログラムを用いて動作を確認する。今回は公式 Web サイトにて `Basic pipeline` として紹介されているサンプル [*2]を実行する。このサンプルは以下の処理を行う。

1. 入力ファイル (sample.fa) の内容を `awk` コマンドで複数ファイルに分割する (1 番目のジョブ)
2. 1 で生成した各ファイルの中身を `rev` コマンドで反転する (2 番目のジョブ)
3. 2 の出力結果を標準出力に表示する

上述の `nextflow.config` ファイルを用意してワークフローを実行すると、1 番目のジョブがバッチジョブとして実行され、その終了を確認してから 2 番目のジョブがバッチジョブとして実行される。TCS において複数のジョブの実行順序を制御したい場合はステップジョブを利用するが、Nextflow を使うことで (ステップジョブを使わずに) ジョブの実行順序制御が可能となる。

入力ファイル sample.fa の中身を図 1 に、実行時の出力情報を図 2、実行後のジョブ実行情報 (TCS の

[*2] `https://www.nextflow.io/basic-pipeline.html`

Listing 1　nextflow.config 設定ファイルの例。各計算ジョブを玄界ノードグループ A にて 4CPU コア最大 10 分の設定で実行する例。

```
process {
  executor = 'tcs'
  time = '00:10:00'
  clusterOptions = '-L rscgrp=a-batch -L vnode-core=4'
}
```

ジョブ実行履歴）を図 3 に示す。入力ファイルの行ごとの文字列が左右逆順で出力されていることと、2 つのジョブが直列に実行されていることが確認できる。このサンプルでは計算ノードのリソースをほとんど利用していないが、複数のジョブの実行順序を制御するワークフローの動作が行えていることは確認できた。

```
[ku40000105@genkai0001 tests]$ cat data/sample.fa
>1aboA
NLFVALYDFVASGDNTLSITKGEKLRVLGYNHNGEWCEAQTKNGQGWVPS
NYITPVN
>1ycsB
KGVIYALWDYEPQNDDELPMKEGDCMTIIHREDEDEIEWWWARLNDKEGY
VPRNLLGLYP
>1pht
GYQYRALYDYKKEREEDIDLHLGDILTVNKGSLVALGFSDGQEARPEEIG
WLNGYNETTGERGDFPGTYVEYIGRKKISP
>1vie
DRVRKKSGAAWQGQIVGWYCTNLTPEGYAVESEAHPGSVQIYPVAALERI
N
>1ihvA
NFRVYYRDSRDPVWKGPAKLLWKGEGAVVIQDNSDIKVVPRRKAKIIRD
```

図 1　ワークフロー実行例 1：入力ファイル

本サンプルは、clusterOptions を適切に設定すれば玄界同様に TCS を利用している他のシステムでも同様に実行可能である。もちろん executor などの設定も変更すれば TCS 以外の環境でも実行可能であり、同じワークフローを多様な環境で容易に利用可能となった。

```
[ku40000105@genkai0001 tests]$ ../nextflow basic.nf

 N E X T F L O W   ~  version 25.07.0-edge

Launching `basic.nf` [jolly_ampere] DSL2 - revision: 464a2e4aaa

executor >  tcs (2)
[c8/eec479] splitSequences [100%] 1 of 1 ✔
[5b/40e9be] reverse        [100%] 1 of 1 ✔
Aoba1>
SPVWGQGNKTQAECWEGNHNYGLVRLKEGKTISLTNDGSAVFDYLAVFLN
NVPTIYN
Bscy1>
YGEKDNLRAWWWEIEDEDERHIITMCDGEKMPLEDDNQPEYDWLAYIVGK
PYLGLLNRPV
thp1>
GIEEPRAEQGDSFGLAVLSGKNVTLIDGLHLDIDEEREKKYDYLARYQYG
PSIKKRGIYEVYTGPFDGREGTTENYGNLW
eiv1>
IRELAAVPYIQVSGPHAESEVAYGEPTLNTCYWGVIQGQWAAGSKKRVRD
N
Avhi1>
DRIIKAKRRPVVKIDSNDQIVVAGEGKWLLKAPGKWVPDRSDRYYVRFN

WARN: [INFO] PJM 0000 pjsub Job 3698478 submitted.
WARN: [INFO] PJM 0000 pjsub Job 3698479 submitted.
```

図 2　ワークフロー実行例 2：実行時の出力（入力ファイルの各行の文字列が逆順になって出力されていることが確認できる）

5　おわりに

本稿ではワークフローシステムを用いたスーパーコンピュータのジョブ実行について述べた。特にバイオインフォマティクス分野などで利用されている Nextflow ワークフローシステムについて、その基本的な機能と、TCS に対応させた実装について述べた。Nextflow はバックエンドとなるジョブスケジューラごとに Executor クラスが明確に分かれており、その実装がシンプルでわかりやすいため僅かな量の実装で Nextflow を TCS に対応させることができた。Nextflow を使えば玄界を含む様々なシステムで同じワークフロージョブを容易に利用可能となり、システム間のワークロードの移植も容易となる。さらに複数システムにまたがるワークフローの実行にも活用でき

```
[ku40000105@genkai0001 tests]$ pjstat2 -H -l
 JOB_ID   JOB_NAME          MD  STATUS  USER        RSCGROUP  START_DATE           ELAPSE    NODE  CORE  GPU  POINT
 3698478  nf-splitSequences  NM  EXT     ku40000105  a-batch   2025/09/23 12:14:00  00:00:03     -     4    -      1
 3698479  nf-reverse         NM  EXT     ku40000105  a-batch   2025/09/23 12:14:08  00:00:01     -     4    -      1
```

図 3　ワークフロー実行例 3：ジョブ実行履歴（START_DATE と ELAPSE の列を見れば 2 つのジョブが直列に実行されたことが確認できる）

る可能性があり、今後のスーパーコンピュータの多様な利用方法を考える上で役立つことが期待される。

謝辞

本研究は文部科学省「次世代計算基盤に係る調査研究」における「運用技術調査研究」の一環として実施されたものです。

参考文献

[1] 平石 拓, 安部 達也, 三宅 洋平, 岩下 武史, 中島 浩, 柔軟かつ直観的な記述が可能なジョブ並列スクリプト言語 Xcrypt, 先進的計算基盤システムシンポジウム（SACSIS2010）, pp.183-191, 2010

[2] P. Di Tommaso, et al. Nextflow enables reproducible computational workflows. Nature Biotechnology 35, 316 – 319 (2017) doi:10.1038/nbt.3820

[3] `https://groovy-lang.org/` (Last accessed 2025-09-23)

付録 A　TCS に対応した Nextflow の入手方法

Nextflow の Web サイト (`https://www.nextflow.io/`) には Nextflow の導入方法として curl コマンドを使ったわずか一行のコマンド (`curl -s https://get.nextflow.io | bash`) が記されているが、この方法では最新のリリース版がダウンロードされる。本稿の執筆時点ではリリース版の Nextflow には TCS に対応する機能がまだ含まれていないため、参考までに各自でコンパイル・ビルドして利用可能にする方法を記しておく。

手順としては、git clone で最新版（もしくは TCS 対応コードが取り込まれた以降の版）のコードを入手し、gradlew コマンドでコンパイル・ビルドをするだけで良い。図 4 に具体的な手順の例を記す。なお`--max-workers=2` は共有のログインノードなどで計算資源を使い過ぎないように最大 2 プロセスしか実行されないよう制限するために付けたオプションである。実行後には`./build/releases` 以下に `nextflow-25.07.0-edge-dist` のような名前の実行可能ファイルが作られるため、これを実行すれば良い。必要に応じて `nextflow` などのわかりやすい名前に変更し、パスの通ったディレクトリに配置すると便利である。

```
$ git clone https://github.com/nextflow-io/nextflow.git
$ cd nextflow
$ ./gradlew compile exportClasspath --max-workers=2
$ BUILD_PACK=1 ./gradlew pack --max-workers=2
```

図 4　TCS に対応した Nextflow の入手とビルドの手順