B-014

ステートフルアプリケーションの Kubernetes クラスタ間マイグレーション方式の設計
Design of Migration Method between Kubernetes Clusters for Stateful Application

土谷 彰義† 出口 彰† 柴山 司†
Akiyoshi Tsuchiya Akira Deguchi Tsukasa Shibayama

1. はじめに

近年，特定環境へのロックインを防ぐため，可搬性の高いアプリケーション（以降，アプリと略す）の実装を可能とするコンテナの利用が広まっている．実行基盤として Kubernetes[1]を利用することで，アプリ運用の一貫性も維持できる．しかし，永続データを使用するステートフルアプリを Kubernetes クラスタ間でマイグレーションするためには，クラスタ間でデータもマイグレーションする必要があり，コンテナのみでは可搬性を確保できない．そこで本稿は，ステートフルアプリのデータも含めてマイグレーションする方式について述べる．

2. 対象システム

対象システムの構成を図 1 に示す．コンテナを用いて実装したアプリを Kubernetes で実行する．複数 Kubernetes クラスタ（以降，クラスタと略す）を Open Cluster Management(以降，OCM と略す)[2]を用いて管理する．OCM は，クラスタへのアプリ配置を管理する．アプリは，ストレージに作成するボリュームに永続データを保存する．

OCM 上でアプリ配置を変更すると，OCM は旧配置クラスタからアプリを削除し，新配置クラスタにアプリを作成することで，アプリをマイグレーションする．

OCM はボリュームが持つデータをマイグレーションする機能を持たない．そこで，本稿では，マイグレーション時に併せてデータもマイグレーションする方式を実現する．

3. 目標

(目標1)操作性：ステートレスアプリと同一

操作の複雑化を避けるため，ステートレスアプリと同一手順でマイグレーション操作可能とすることをめざす．

(目標2)ダウンタイム：300 秒（5 分）未満

アプリ稼働率はビジネス上重要であるため，マイグレーションに伴うアプリ停止によって稼働率要件に違反することがない方式の実現をめざす．稼働率 99.99%以上が要件であるクリティカルアプリの年間許容ダウンタイムは 52 分である[3]．マイグレーション以外の要因による停止も考慮し，年間許容ダウンタイムの 10%(5 分)未満を目標とした．

4. 課題

(課題1)データコピー操作の自動化

手動操作によるデータコピーによってアプリをマイグレーションすることは可能であるものの，(目標 1)を満たさない操作性となる．したがって，OCM でのアプリ配置変更操作に連動し，データを自動コピーすることが課題となる．

(課題2)データコピーに伴うダウンタイム短縮

旧配置クラスタでアプリがボリュームに書き込んだデータを漏れなくコピーするため，アプリ停止の上でデータコピーする方法が考えられる．この方法では，大サイズのデータコピーの場合，ダウンタイムが長時間化し，(目標 2)を満たせない．例えば，10Gbps でコピーする場合，100GiB では約 1.4 分，1TiB で約 15 分をコピーに要する．したがって，データコピーに伴うダウンタイムの短縮が必要となる．

† 株式会社 日立製作所 研究開発グループ，Hitachi, Ltd., Research & Development Group

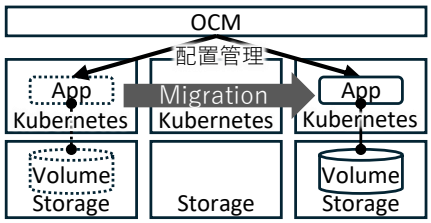

図 1 対象システム

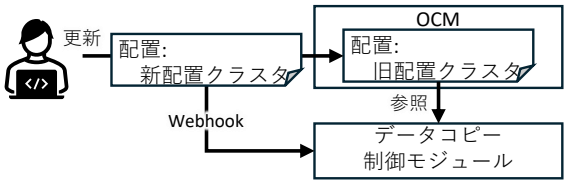

図 2 アプリ配置変更検知

5. 提案方式

5.1 アプリ配置変更検知

OCM でのアプリ配置変更操作に連動して自動的にデータコピーを開始するため，当操作を検知する必要がある．この方法を図 2 に示す．ユーザがアプリ配置変更を OCM にリクエストすると，データコピー制御モジュールは，当リクエストを Webhook し，マイグレーション操作，および新配置クラスタの情報を検知する．加えて，当モジュールは，OCM が持つ配置情報を参照し，旧配置クラスタの情報を得る．両情報を基に，当モジュールは，次節で述べる制御方式を用い，旧配置クラスタが使用するストレージから新配置クラスタが使用するストレージへデータをコピーする．

5.2 データコピー制御方式

ダウンタイムを短く抑えつつデータコピーする方式を図 3 に示す．まず，アプリを実行したままコピーを開始するため，ボリュームに書き込んだデータを継続的にコピーする技術を用いる[4]．当技術は，最初にボリュームの全データをコピーし（以降，初期コピーと呼ぶ），その後にアプリが更新したデータを継続的にコピーする（以降，更新コピーと呼ぶ）．これにより，アプリがデータをボリュームに書き込み続けても，漏れなくデータをコピーできる．

5.1 節で述べた方法によりアプリ配置変更を検知した後，以下の手順で当コピー技術を自動操作することで，ダウンタイムを短く抑えつつ，データをマイグレーションする．

① 旧配置クラスタでアプリを実行したままコピーを開始し、コピーの進捗状況を監視

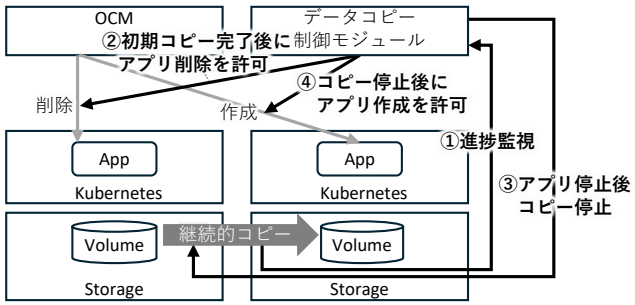

図 3　データコピー制御

表 1 マイグレーション手順の比較

	ステートレスアプリ	ステートフルアプリ
ユーザ操作	アプリ配置変更	アプリ配置変更
内部制御	- 旧/新配置クラスタへのアプリ削除/作成	- 旧/新配置クラスタへのアプリ削除/作成 -データコピー開始/停止

② 初期コピー完了後，OCM による旧配置クラスタでのアプリ削除を許可
③ アプリ停止後，未コピーデータの更新コピー完了を待ち，コピー停止
④ コピー停止後，新配置クラスタへのアプリ作成を許可

ステップ①②により，大サイズデータの場合に長時間化する初期コピーの完了後にアプリ停止することで，初期コピー中のアプリダウンを避けることができる．また，ステップ③④により，コピー漏れのデータが存在する状態で新配置クラスタにアプリ起動することを防止する．

更新コピーが必要なデータ量に応じて，ステップ③の更新コピーの完了待ち時間は長くなる．しかし，このデータ量はボリューム全体のサイズよりは十分小さいと考えられるため，本方式により，ダウンタイムを短縮可能である．

6. 机上評価

6.1 操作性

表 1 にマイグレーションのユーザ操作と内部制御を示す．ステートレスアプリに対しては，ユーザはアプリ配置変更操作のみでマイグレーションを実行できる．ステートフルアプリも同様に，ユーザはアプリ配置変更の操作でマイグレーションを実行でき，必要なデータコピーは内部で自動制御される．したがって，ステートレスアプリとステートフルアプリを同一操作でマイグレーション可能といえる．

6.2 ダウンタイム

6.2.1 見積方法

提案方式のダウンタイム期間を図 4 に示す．提案方式のダウンタイム(T_{down})は，以下の式で見積もることができる．

$$T_{down} = T_{term} + T_{copy} + T_{launch}$$

ダウンタイム(T_{down})は，アプリ停止処理時間(T_{term})，アプリ停止時点の未コピーデータの更新コピー時間(T_{copy})，およびアプリ起動処理時間(T_{launch})からなる．アプリが停止を始めると，起動完了まで当該アプリを利用できないため、

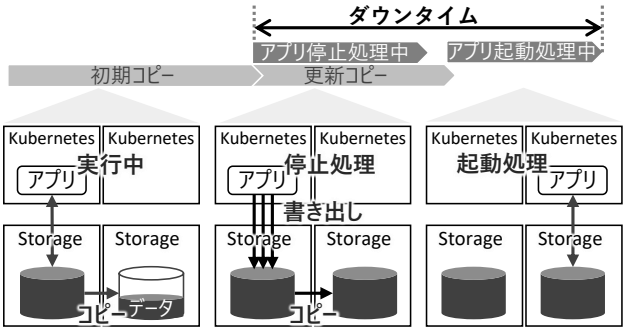

図 4 状態遷移とダウンタイム

表 2 見積もり結果

項目		見積もり
T_{down}		257 秒
	T_{term}	23 秒
	T_{copy}	230 秒
	T_{launch}	4 秒

これら 3 つの時間がダウンタイムに含まれる．

6.2.2 見積もり結果

見積もり結果を表 2 に示す．Kubernetes のコンテナ管理単位である POD の削除時間が長くとも約 23 秒であるため[5]，T_{term} を同時間と推定した．また，クラスタとストレージが 10Gbps で接続され，アプリ停止処理が T_{term} の間データをボリュームに帯域限界性能で書き込み，このデータを 1Gbps のストレージ間帯域でコピーしたことを想定し，T_{copy} を 230 秒と推定した．最後に，Kubernetes の POD 起動時間が約 4 秒であるため[6]，T_{launch} を同時間と推定した．

これらを基にダウンタイム（T_{down}）を見積もると，257 秒となり，目標を満たす．

7. おわりに

本稿では，OCM と Kubernetes を用いてコンテナアプリを運用するシステムを対象に，データとともにステートフルアプリをマイグレーションする方式について述べた．アプリの配置変更操作を検知して自動的にデータコピーすることで，ステートレスアプリと同一操作でのマイグレーションを実現した．加えて，アプリを実行したままボリュームの初期コピーを実行するコピー制御により，ダウンタイムの長時間化を避ける．ダウンタイムの机上評価により，クリティカルアプリの運用に耐えうる 257 秒まで短縮できる可能性があることを示した．今後，実機を用いた評価により，提案方式を評価する必要がある．

参考文献

[1] https://kubernetes.io/
[2] https://open-cluster-management.io/
[3] A. Berenberg, B. Calder. “Deployment Archetypes for Cloud Applications”, ACM Computing Surveys, Vol. 55, Issue 3, Article 61 (2022)
[4] M, Ji, A. Veitch, J. Wilke, “Seneca: Remote Mirroring Done Write”, Proc. of USENIX Annual Technical Conference (2003)
[5] https://docs.openstack.org/developer/performance-docs/test_results/container_cluster_systems/kubernetes/API_testing/index.html
[6] https://kubernetes.io/blog/2015/09/kubernetes-performance-measurements-and/