2K-08

Linuxコンテナ稼働環境のための斬新な資源管理

奥村開里 † 吉瀬謙二 †
† 東京工業大学大学院計算理工学専攻

1 はじめに

近年、仮想化環境を用いてサーバー統合を行う際に、仮想マシンを用いて複数の OS を稼働させる方法や、コンテナを作成し、OS のリソースを分割する方法などが取られている。

コンテナを用いてサーバー統合を行った際には、仮想マシンを用いた場合と比べ、仮想マシンのオーバーヘッドが存在しない為、高集積なサーバー統合を行える。しかし、統合された複数のサーバーからのリクエストにより I/O 資源の競合の問題も大きくなり、SSD などの高スループットなストレージを用いたとしても、I/O が大量のデータを処理する際のボトルネックになってしまう。

また、コンテナ毎に I/O の優先度を割り当て、その割り当てた優先度に応じ、コンテナごとのパフォーマンスを変化させるという機能も必要とされている。しかし、現在 Linux に組み込まれているスケジューラでは、SSD の環境において、高スループットを達成する適切なスケジューリングと、優先度を変化させる機能の両立ができていないという問題がある。

そこで本論文では、コンテナによるサーバー統合環境において、I/O 資源の強豪により、I/O がボトルネックとなっている問題と、コンテナ毎のパフォーマンスを変化させる機能の両立ができていないという問題を解決するために、コンテナ稼働環境での高スループットかつ、コンテナごとの公平性と優先度の変更を達成するフラッシュストレージのための I/O スケジューラを提案する。

2 研究の背景

この節では、コンテナの特徴及び、SSD の特徴とその特徴を活かした I/O スケジューラについて述べる。

2.1 コンテナ

コンテナ自体は古くから使われている技術であるが、近年、コンテナ技術を利用してアプリケーションのデプロイメントの自動化を行う Docker[1] や、クラウドコンピューティングプロジェクトである OpenStack[2] でもコンテナがサポートされたこともあり、IaaS としての運用も考えられており、再注目されている技術である。

コンテナはプロセスグループのリソースの利用を制限・隔離する cgroups というカーネルの機能と、名前空間を提供する namespace というカーネルの機能 2 つを組み合わせて作られる OS 上の分離空間である。その為、仮想マシンによってサーバー統合を行った場合とは異なり、ホストからはゲスト上のプロセスが見え、プロセス、プロセスグループごとの細かい制御が容易であるという特徴がある。

2.2 SSD の特徴

SSD のスケジューリングを行う際に考慮すべき特徴として、

- 読み込みと比べて書き込みのコストが大きいこと
- シーク時間が存在しないこと
- リクエストのサイズとリクエストにかかる時間に相関があること

などが挙げられ、SSD を使用する際には SSD のこれらの特徴を活かしたスケジューリングを行うべきである。

しかし、現行の 3.x 系の Linux Kernel に組み込まれているスケジューラには、Completely Fair Queueing(CFQ) や Deadline、Noop などがあるが、これらは HDD のシーク時間の削減を念頭においたスケジューラであり、SSD のパフォーマンスを十分に引き出せていない。

2.3 先行研究

すでに、上記の SSD の特徴を活かしたスケジューラの研究は行われている。[3][4][5] 先行研究 [3] で提案されている IRBW スケジューラでは、Read リクエストと Write リクエストを区別し、Write リクエストについては論理ブロックサイズにまとめると（LB バンドリング）、SSD のスループットが向上するという結果が出ている。

また、さらに先行研究 [4] で IRBW スケジューラを元に提案されている STB スケジューラでは、Write リ

Innovative Resouce Management for Linux Container Environment
Kairi OKUMURA†and Kenji KISE†
†Department of Computer Science, Tokyo Institute of Technology
okumura@arch.cs.titech.ac.jp, kise@cs.titech.ac.jp

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.

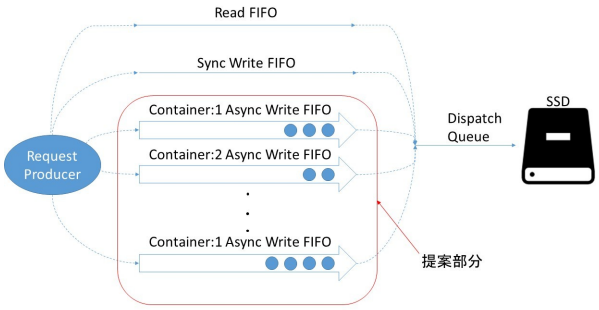

図 1: 提案するスケジューリングポリシー

クエストを更に同期I/Oと非同期I/Oに分類し、Writeの Asynchrounous リクエストについてはLBバンドリングを行い、Synchronous I/O は即座にディスパッチを行うというスケジューリングポリシーによってさらなる性能の向上が達成されている。

そこで本論文で提案する手法においても、このポリシーを元にする。

3 提案手法

この節では、我々の提案したSSDの特徴とコンテナの特徴を考慮したI/Oスケジューラの具体的な構想を示す。我々の提案したスケジューリングポリシーでは、STBスケジューラにおいて、コンテナごとにLBバンドリングを行うキューを持たせることによって、I/O資源の分配の偏りを防ぎ、コンテナ毎の優先度の設定を容易にする。

具体的には、次のようなポリシーでスケジューリングを行う。

- 先行研究 [4] でのポリシーを利用した部分
  - SSDにおいてReadの速度は一定なのでReadリクエストは即座にディスパッチ
  - 同期Writeリクエストは即座にディスパッチ
  - 非同期WriteリクエストについてはLBバンドリングを行う
- 提案部分
  - 同一コンテナに所属するプロセスグループ毎にLBバンドリングの為のキューとdeadlineをもたせる
  - LBを超えていて、かつ優先順位が高いものからディスパッチ

このポリシーのイメージ図を**図 1** に示す。

コンテナというプロセスグループごとにキューを持たせることによって、プロセス毎にキューを振り分けた際に問題となるディスパッチするべきキューの探索コストの問題と、キューを振り分けなかった際に問題となる公平性の低さの問題を同時に解決できる。

また、LBバンドリングはそれぞれのコンテナ（プロセスグループ）が一台のサーバーに相当するI/Oを必要とすると考えられ、本提案手法においても、適切に機能すると考えられる。

また、キュー毎にdeadlineを設定することによって、LBバンドリングの効率だけではなく、許容範囲内のレスポンスタイムを達成できる。

このポリシーによって従来の手法で問題となっていた公平性の低さと、SSDの特徴を活かしSSDの性能を引き出すスケジューリングを行う。

4 今後の方針

今後の方針として、本論文で提案したI/Oスケジューラは CFQ で実装されている Hierarchical Group Scheduling[6] と、STBスケジューラを参考にC言語でカーネルモジュールとして実装を進めていく。また評価方法として、スループットには、既存のFIOベンチマークを使用し、公平性の測定には、並列に動いているコンテナのIOPS(I/O per Second)、もしくはタイムスライスの割合によって測定を行おうと考えている。また、キュー毎の優先順位の計算をIOPSに基づくか、タイムスライスに基づくかを実験を行って決定する。

参考文献

[1] Docker:https://www.docker.com/

[2] OpenStack:http://www.openstack.org/

[3] Kim, Jaeho, et al. "Disk schedulers for solid state drivers." Proceedings of the seventh ACM international conference on Embedded software. ACM, 2009.

[4] Kang, Seungyup, Hyunchan Park, and Chuck Yoo. "Performance enhancement of I/O scheduler for solid state devices." Consumer Electronics (ICCE), 2011 IEEE International Conference on. IEEE, 2011.

[5] Park, Stan, and Kai Shen. "FIOS: a fair, efficient flash I/O scheduler." FAST. 2012.

[6] Hierarchical group I/O scheduling:http://lwn.net/Articles/427961/

[7] FIO:http://freecode.com/projects/fio

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.