2009－OS－110 (7)
2009／1／29

LB-CAS(LoopBack Content Addressable Storage)を用いた OS 起動時のアクセス解析

八木 豊志樹　　須崎 有康
産業技術総合研究所　情報セキュリティ研究センター

我々は仮想マシンを用いた OS Circular の開発を行っている。OS Circular は仮想マシンとインターネット仮想ディスク LB-CAS(LoopBack Content Addressable Storage)からなり、インターネット上の HTTP サーバにあるデータファイルをダウンロードしてマップすることによって仮想ディスクを構築している。仮想化するレイヤが増えると、それだけレイヤ間でのセマンティックギャップが生じる。今回我々は LB-CAS に対する性能評価を行い、実際に発生するセマンティックギャップの測定を行った。

Access analysis on OS boot using LB-CAS (Loop-back Content Addressable Storage)

Toshiki Yagi　　Kuniyasu Suzaki
AIST　RCIS

We developed “OS Circular” which uses the virtual machine. OS Circular consists of the virtual machine and Internet virtual disk “LB-CAS(LoopBack Content Addressable Storage)”, which downloads a data file from HTTP server on Internet and maps it on a virtual disk. When the virtualization layer increases, the semantics gap between the layers occurs. In this paper we measure the performance of LB-CAS and make clear the semantics gap which really occurs.

1. はじめに

我々は仮想マシンを用いた OS Circular の開発を行っている。OS Circular では仮想マシンがインターネット仮想ディスク LB-CAS(LoopBack Content Addressable Storage)を起動デバイスと利用する。

CAS は USENIX FAST 04[1]で発表され、現在もデータの新しい管理技術として拡張が行われている。CAS では物理アドレスではなくコンテンツの SHA1 ダイジェストでブロック管理する。異なる物理アドレスに同じデータがある場合は同じものと管理されるため総ストレージ容量を削減できる。

また、外部フラグメンテーションが起こらないことやSHA1が異なる追記管理のためロールバックが可能であるという特徴を持つ。我々はCASを仮想ディスクに適用したLB-CASを作成し、OS起動を可能にした研究をUSENIX LISA07[2]において発表した。ローカルなハードディスクはInternet上のCASデータのキャッシュとして働き、必要なデータをキャッシュしていればネットワークが使えないモバイル環境でもOS起動が可能である。

CASベースの仮想ディスクでは「ファイルシステムとCASブロック間（フラグメンテーション問題）」および「CASブロックとローカルキャッシュ（再ダウンロード問題）」のセマンティックギャップがあることが判っている。本論文ではこの問題を明確にするとととともにその対処と性能を示す。

2. LB-CASのセマンティックギャップ

CASベースの仮想ディスクでは2つのセマンティックギャップにより不要なデータが転送されることが判っている。一つは「ファイルシステムとCASブロック間のセマンティックギャップ」であり、もう一つは「CASブロックとローカルキャッシュのセマンティックギャップ」である。

ファイルシステムとCASブロック間のセマンティックギャップはファイルシステムが想定するブロックサイズ(4KBが標準)とCASが想定するブロックサイズ(256KB)の違いから発生するギャップである。古くからあるフラグメンテーション問題と同一だが、CASでは容量のロスよりアクセス効率が問題になる。CASブロックデータ内で実際使われるデータの割合が低いとSHA1管理のオーバーヘッドが顕在化する。このためできるだけ1つのCASブロックで利用されるデータの割合を高める必要がある。この問題解決にはファイルシステムとブロックデバイスの配置確認ツールや再配置ツール(Windowsの Perfect Diskや Disk Keeperなど)を適用する必要がある。

CASブロックとローカルキャッシュのセマンティックギャップは、CPUのキャッシュと同じ問題である。つまり、必要なデータのみを効率的にダウンロードしてキャッシュのヒット率を上げ、再ダウンロードを防ぐことで高速なブロックデバイスにする。この最適化にはユーザの利用履歴から使われるCASブロックを推定してあらかじめキャッシュする方式が有効である。同様の手法はLinuxカーネルのreadahead[3]で利用されており、一回のディスクアクセスで効率的にデータがメモリにキャッシュされている。

今回我々はOS Circularに対する性能評価を行い、実際に発生するセマンティックギャップの測定を行った。

3. 性能測定

今回、我々はファイルシステムとCASブロック間のセマンティックギャップを、ext2ファイルシステムにてDebian Lenny (dd, tar)およびKnoppix 4.0.1 (boot)がインストールされたディスクイメージに対し、0ms, 25ms, 50ms, 100msの遅延を入れた状態にてネットワークに生じたトラフィックを計測した。

3.1. 測定に用いた環境

今回測定に用いた機器は以下のとおりである。

サーバ

- ThinkPad T60
  - Intel Core Solo (T1300 1.66GHz)
  - 2.0GB RAM
  - 60GB SSD
  - Intel 82573L GbE controller
  - Ubuntu 8.10 Server (kernel 2.6.27-9-server)
  - Apache Web server 2.2.9

クライアント

- ThinkPad X60
  - Intel Core2 Duo(T7200 2GHz)
  - 2.0GB RAM
  - 60GB SSD
  - Intel 82573M GbE controller
  - Knoppix 5.3.1(kernel 2.6.24.4)

3.2. LB-CAS の LAN 環境における性能評価

LB-CAS は、インターネット上の HTTP サーバから必要な CAS ブロックをオンデマンドでダウンロードし、それを展開することにより、仮想的にディスクとして見せる仮想ストレージシステムである。LB-CAS では、CAS ブロックを一定のサイズごとに区切ってダウンロードしている。ブロックサイズが大きくなれば、ダウンロードのスループットは向上するが、本来ファイルシステムが必要とするデータの容量より多くのデータをダウンロードする必要がある。

図 1・図 2 は、ブロックサイズをそれぞれ 64kB, 128kB, 256kB, 512kB としてディスクイメージに dd をかけた時の読み込みサイズ総量およびスループットである。dd はデバイスの先頭から順にすべての領域に対して読み出しを行うため、スループットが高いほど処理が速く終了している。

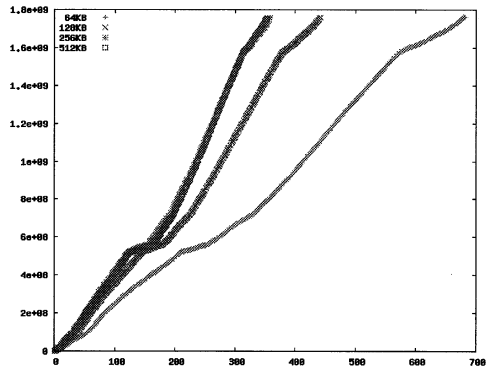

図 1　総転送量の推移(dd)

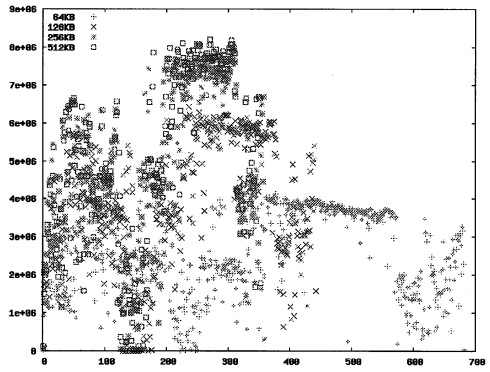

図 2　スループットの推移(dd)

図 3・図 4 は、同条件においてディスクイメージの中にあるファイルを、tar を用いて読み出した時の読み込みサイズ総量およびスループットである。また、図 5 が tar を用いた時にディスクイメージに発生したアクセス要求である。tar で読み出しを行った場合もスループットはブロックサイズが大きくなるにしたがって速くなっている。しかしながら、必要とする CAS ブロックの

総量が CAS ブロックサイズが大きくなるにしたがって増加している、つまり必要とするデータと実際に読み込まれるデータの差が大きくなるため、処理が完了するまでの時間は逆転を起こしている。

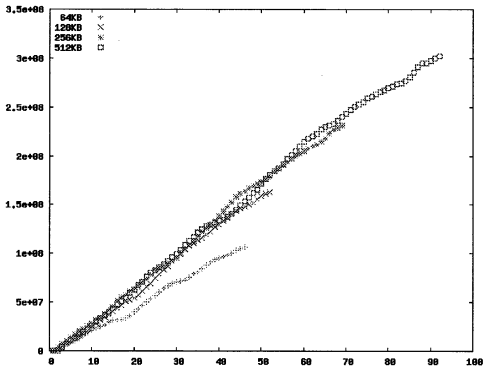

図 3　総転送量の推移(tar)

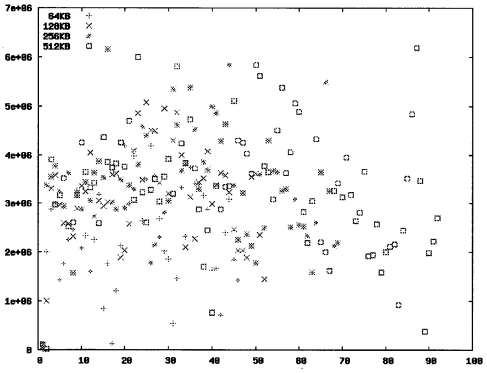

図 4 スループットの推移(tar)

表 1にリクエストが起こった総容量と実際に読み込まれた展開済みブロックの総容量、読み込みサイズのうちどの程度利用されたかを表す充填率を示す。Ext3 ファイルシステムの標準ページサイズである 4kB に比べ 16 倍の大きさである 64kB の CAS ブロックで充填率が 33%程度であり、ブロックサイズを大きくするほど充填率が下がっていく、つまり使われない領域が増加していることが分かる。

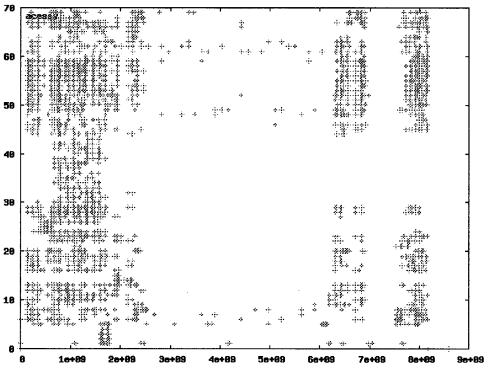

図 5 ext3 ファイルシステムに tar を実行した際のブロックデバイスへのアクセス要求

	要求サイズ	読込サイズ	充填率
64kB	107065344	327614464	32.68%
128kB	106467328	460849152	23.10%
256kB	107769856	629145600	17.13%
512kB	109031424	818413568	13.32%

表 1　tar　にて要求のあったサイズと読み込まれたブロックの差異

図 6がそれぞれのブロックサイズで起動を行った時の読み込み総量である。ここでも tar と同様の傾向が見られる。

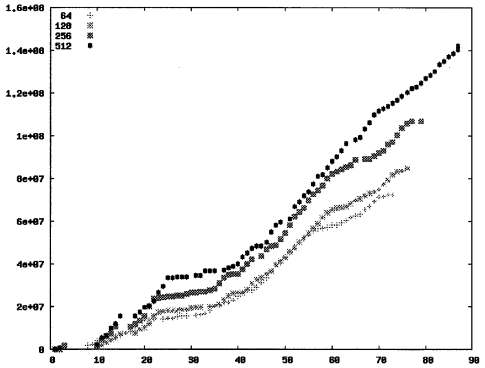

図 6 総転送量の推移(knoppix4.0.2 boot)

3.3. ネットワーク遅延を挿入した場合の性能評価

LB-CAS はネットワーク上で動作するため、ネットワークのバンド幅あるいは遅延の影響を受ける。そのため、先述したブロック充足率による小さなブロックサイズの利点を、スループット差によって吸収されてしまう。図 7〜図 12 に dd を用い、遅延を 25ms, 50ms,100ms に取った場合の総転送量およびスループットのグラフを示す。

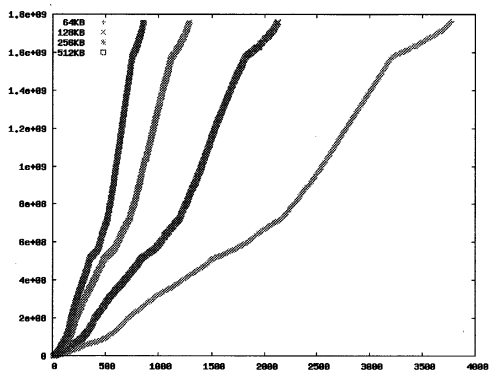

図 7 総転送量の推移(25ms, dd)

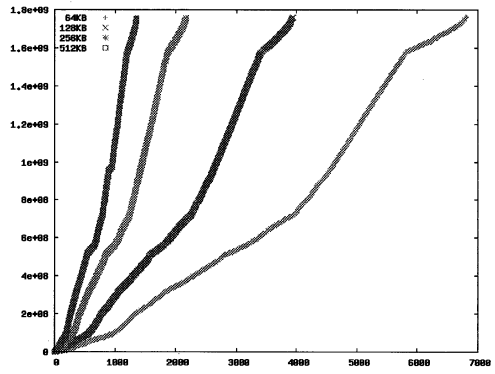

図 8 総転送量の推移(50ms, dd)

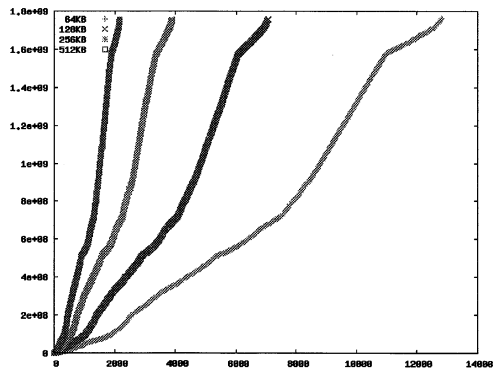

図 9 総転送量の推移(100ms, dd)

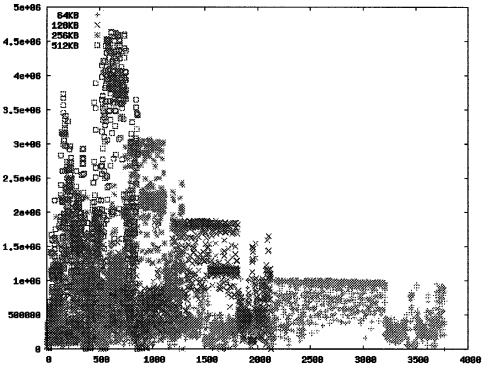

図 10 スループットの推移(25ms, dd)

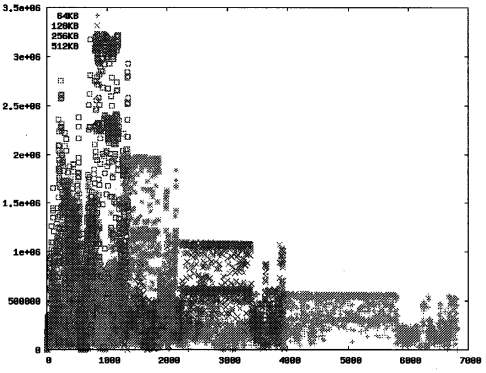

図 11 スループットの推移(50ms, dd)

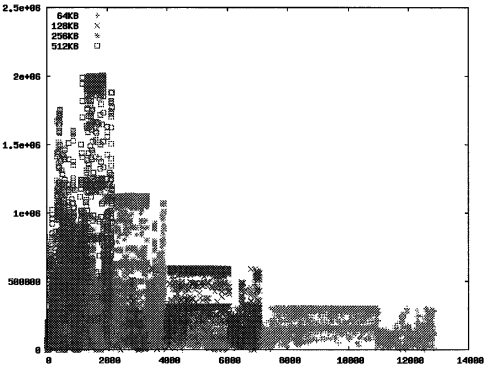

図 12 スループットの推移(100ms, dd)

ネットワーク遅延を挿入することにより、スループットに悪影響が出ていることが分かる。また、この悪影響は小さな転送単位(CAS ブロックのサイズ)において顕著に出ていることが確認される。

また、tar を用いて同様の計測をした場合について、25ms, 50ms, 100ms の遅延を入れた場合の総転送量およびスループットのグラフを図 13～図 18 に示す。

このように、遅延を挿入するとブロック

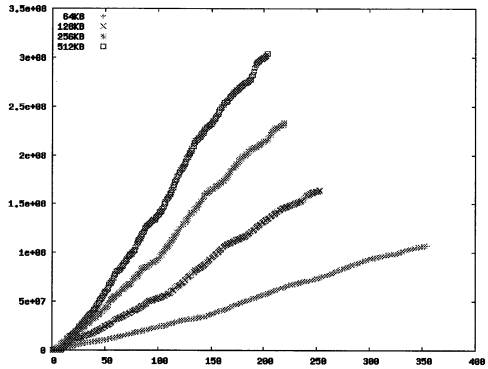

図 13　総転送量の推移(25ms, tar)

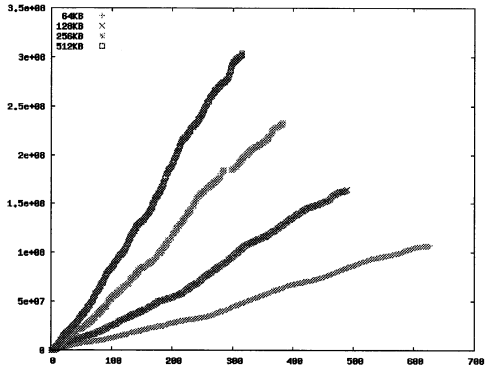

図 14 総転送量の推移(50ms, tar)

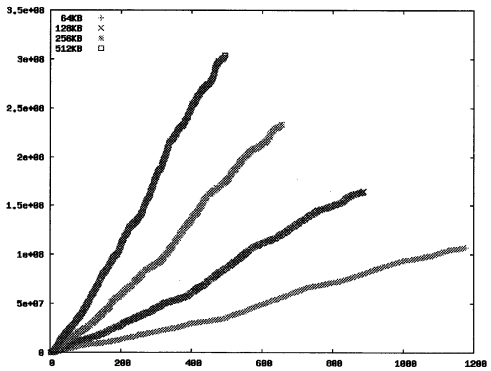

図 15 総転送量の推移(100ms, tar)

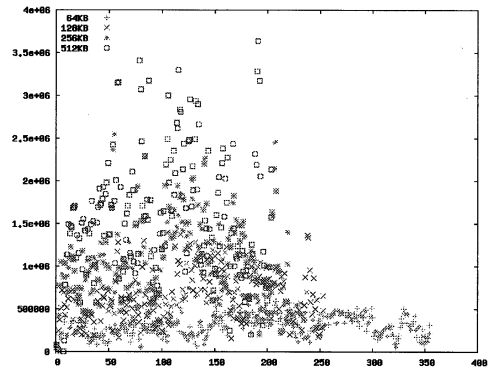

図 16 スループットの推移(25ms, tar)

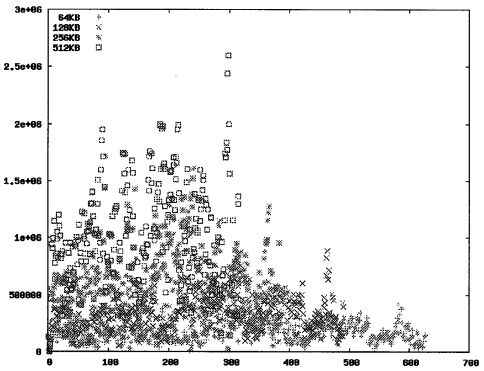

図 17 スループットの推移(50ms, tar)

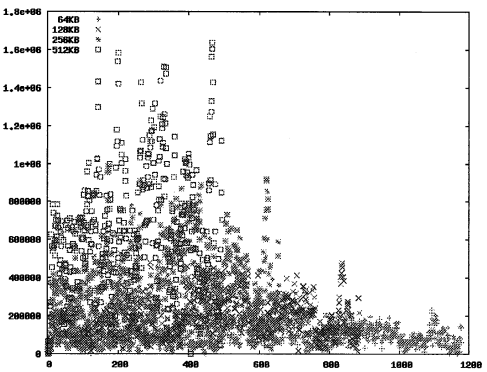

図 18 スループットの推移(100ms, tar)

サイズ 64kB における高充填率の利点が、転送単位の小ささから来るスループット低下の影響を吸収しきれなくなる。このため、充填率を犠牲にしてブロックサイズをある程度大きくした方がインターネット環境で利用する LB-CAS のようなアプリケーションには適していると言える。

4. 諸問題の解決法

インターネットで生じる遅延を考慮に入れると、転送の単位である CAS ブロックサイズを大きくした方が効率がよい。しかしながら、CAS ブロックサイズを大きくすると、CAS ブロックとその中にあるファイルシステムとのシマンテックギャップにより、CAS ブロックの利用率が下がってしまう。ファイルシステムが効率よく利用されれば、その問題は解決される。北川らにより開発されている Ext2optimizer[4]により、ブート時に限定されるが、CAS ブロックの充足率を高めることができる。図 19 に Ext2optimizer を適用前後のブート時における総転送量の推移を示す。

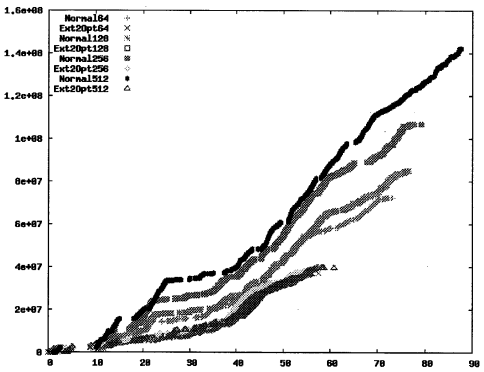

図 19 Ext2optimizer を適用した場合の転送量の推移

上の 4 本のグラフが Ext2optimizer 適用前、下のほぼ重なった 4 本のグラフが適用後の転送量推移となる。Ext2optimizer を適用することにより、ブート時にダウンロードする CAS ブロックの充填率が高まり、ブートに要する時間が減少している。図 20 に遅延を挿入した場合における Ext2optimizer の効果を示す。CAS 充填率が高まることにより、遅延を挿入した場合においても起動時間の短縮が生じていることを確認した。

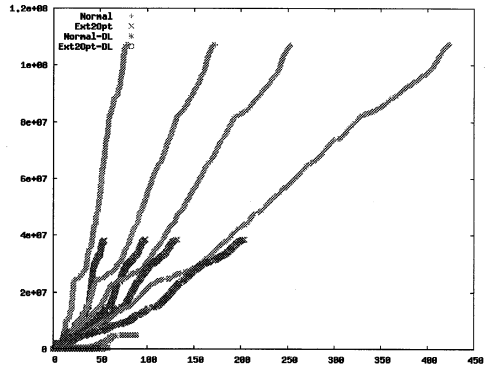

図 20　0ms, 25ms, 50ms, 100ms の遅延を挿入した場合のブート時における総転送量の推移

Ext2optimizer は実際に行ったファイルシステムへのアクセスのログから、アクセスのあったページをブロックデバイスの先頭に再配置する実装となっている。このため、OS のブートのような連続するディスクアクセスには容易に適用できる。その一方でユーザ操作によるディスクアクセスに対応するのは困難である。

5. まとめ

今回我々はLB-CASに対する性能評価を行い、セマンティックギャップが性能に与える影響を示した。CASブロック利用率を高めるために CAS ブロックサイズを小さくすると、遅延の発生するインターネットでの利用ではスループットの低下につながり性能が低下することが分かった。

解決策の一つとして Ext2optimizer を挙げ、これがブート時における性能改善に貢献していることを示した。今後この機構を改良し、より一般的なアプリケーションの起動においても高速化を行いたい。

6. 参考文献

[1] S.Quinlan and S.Dorward, "Venti: A New Approach to Archival Storage," Proceedings of the 1st USENIX Conference on File and Storage Technologies, Monterey, CA, January, 2002.

[2] K.Suzaki, T.Yagi, K.Iijima, and N.A.Quynh, "OS Circular: Internet Client for Reference", Proceedings of the 21st Large Installation System Administration Conference. pp.105–116, Dallas, TX, November, 2007.

[3] WU Fengguang, XI Hongsheng, and XU Chenfeng , "On the design of a new Linux readahead framework", ACM SIGOPS Operating Systems Review, Volume 42, Issue 5, pp 75-84, July, 2008.

[4] K.Kitagawa, H.Tan, D.Abe, D.Chiba, K.Suzaki, K.Iijima, and T.Yagi, "File System (Ext2) Optimization for Compressed Loopback Device",13th International Linux System Technology Conference, pp.25-33, Nürnberg, Germany, September, 2006