Vol.2012-IOT-16 No.14
2012/3/15

ネットブート環境における読み込みキャッシュ機構の改善による起動時間短縮の試み

八田直樹[†1] 丸山伸[†1] 松川正義[†2]
西村浩二[†2] 相原玲二[†2]

近年，ネットブート環境におけるブートサーバへの負荷軽減のために，端末側に読み込みキャッシュを設けることが有効であると言われている．この方式により端末のハードディスク内にサーバ上の仮想ディスクをキャッシュすることでサーバへのアクセスを減らし，多数の端末を一斉に起動した場合においても起動時間を安定化できる．しかし，ハードディスクは一般にランダムアクセスの性能が劣るため，アクセスが多発するとキャッシュを利用しない場合に比して起動時間が延びることが問題となっていた．

そこで本研究では，起動時のアクセスがほぼシーケンシャルなアクセスとなるように端末内のキャッシュの構造を改善することで起動時間を短縮した．そしてキャッシュへのアクセスにプリフェッチ機構を追加することにより起動処理を短縮できることを確認した．

Attempt to reduce startup time by improving the read cache mechanism of netboot environment

Naoki Hatta,[†1] Shin Maruyama,[†1]
Masayoshi Matsukawa,[†2] Kouji Nishimura[†2]
and Reiji Aibara[†2]

Recently, read cache system for network booting system is used widely to reduce the load of the boot server. The read cache system caches parts of the virtual disk on server into the clients' local hard disk, and it stabilizes the start-up time of clients even when a large number of them start booting all at once. However, the performance of hard disk is poor for random access in general, and start-up time using read cache system is extended as compared to that without cache system.

In this study, we improved the structure of the cache in clients' local hard disk so that the access pattern during startup becomes sequential access, and reduced the startup time. We also added prefetch mechanism to the system and confirmed that the startup time can be shortened more.

1. はじめに

多数の端末を保有する時，それぞれの端末のディスクイメージを個別に管理するコストは非常に高い．そこでネットブートにより端末内にハードディスクを持たないようにすることで，これまで端末ごとのディスクの管理だったものをサーバ上にあるディスクイメージの管理へと転換するネットブート技術に注目が集まっている．各端末内の HDD に情報が分散させるのではなく，サーバ上に各端末ごとのディスクイメージを保持することで，ディスク内容の更新や復元が容易になった．また，ネットブート環境においては，端末が利用するディスクイメージを切り替えるだけで，時間帯や利用者の選択によりディスクイメージを切り替えることを容易に行える．

さらに，端末からの書き込み要求に対してディスクイメージを書き換えるのではなく，端末ごとの書き換えを管理する機構「Write Cache」を導入することで，端末それぞれに対応したディスクイメージを持つのではなく，1 つのディスクイメージを多数の端末で共有できるようになる．この Write Cache 機構により多数の端末のディスクイメージの一元管理が容易となり，教育用計算機やマンガ喫茶といった「多数の端末を共通のディスクイメージで動作させる環境」ではネットブートが広く利用されている．

Write Cache 機構を備えたネットブートシステムでは，1 つのイメージを管理するだけですべての端末のディスクを管理できるというメリットがある．その反面，サーバ上に置かれた 1 つのディスクイメージを多数の端末で同時に利用することから，多数の端末が一斉に起動するなどして同時にサーバにアクセスした際，サーバの負荷により端末の動作が遅くなるという問題があった．この問題への対策として「1 台のサーバを利用する端末数を適正な台数（一般には 20-30 台）に抑える」という対策もあるが，これは端末数の多い環境ではサーバ数が増大することになり多数のサーバを管理するコストの増大という問題を生む

†1 株式会社シー・オー・コンヴ
CO-CONV, Corp
†2 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University

ことになる．そこでネットブート端末のそれぞれにハードディスクを備え、そのディスクの一部を読み込みキャッシュとして利用することでサーバへの負荷を下げる “Read Cache 技術” が注目されている．この技術により，1 台のサーバを数百台の端末で利用しても適切に動作するようになった．

しかし，ReadCache 技術によるキャッシュデータは端末内のハードディスクに蓄積されるため，ネットブート環境でありながらハードディスクへのアクセスが多発することになる．また，ハードディスクは Write Cache 機構によるキャッシュ領域や OS によるワークエリアとしても利用されるため，端末起動時にはハードディスクへのアクセスやヘッドシークが特に多く発生し，結果として端末の起動に時間がかかるという問題が生じていた．

そこで，本研究では，ReadCache のキャッシュデータをハードディスクに保持する際，キャッシュデータ専用のデータ領域を確保しつつ，読み込み要求があった順にキャッシュ領域の先頭から保存していく方式を採用した．このような形式にすることで，起動時に読み込まれるデータはキャッシュ領域の先頭部分に蓄積されることになる．そして端末起動時にはキャッシュ領域の先頭部分を一括してメモリ上に読み込むことで，それ以降の読み込み要求に対してハードディスクにアクセスすることなく迅速に応答できるようになることを確認した．

以下，2 章ではネットブートシステムの特徴と従来の各種研究や実装方式の課題について述べる．3 章で本提案の特徴，4 章では提案方式の実装について述べる．そして本提案方式を実装し評価した結果について 5 章にまとめる．最後に本研究のまとめと今後の課題について 6 章で述べる．

2. 従来の研究

2.1 ディスク管理ソリューションとしてのネットブート方式

端末が起動し動作する際に利用する記憶装置として通常は端末内にあるハードディスクを利用する．また通常の端末ではハードディスク内のデータは端末の動作に伴い自由に書き換えられるため，端末が多数存在する時にはそれぞれのハードディスクは内容が異なる状態となってしまう．このような状態では多数の端末を安定して動作させることは難しく，各端末のハードディスクの状態を一定の状態に維持するための「ディスク管理ソリューション」が求められるようになった．このようなディスク管理ソリューションの１つとして,「ネットブート方式」を活用したディスク管理手法が注目されている．

ネットブート方式のシステムでは，端末内にあるハードディスクの内容をイメージファイルに変換しネットブートサーバ上に配置する．そして各端末はこのファイルの内容を利用して起動するように設定される．このように設定することで，イメージファイル１つに対して端末 1 台が起動するネットブートシステムとなる．さらに，端末からの書き込み要求に対してイメージファイルを書き換えるのではなく，変更された内容を “Write Cache” として端末ごとに異なるファイルとして保持することで 1 つのイメージファイルを用いて多数の端末を動作させられるようになる．そして端末の再起動時に “Write Cache” の内容を初期化することで，毎回一定のディスクイメージから動作を開始することが出来る．

このようにネットブート方式は多数の端末を一定のディスクイメージで動作させるために適している．しかし，端末が起動する際には多数のプログラムやファイルを読み込むため，ネットブートサーバに通常よりも高い負荷をかける．また，多数の端末を一斉に起動した際にはその負荷がサーバに集中し，端末の起動に非常に長い時間がかかることになる．特に教育機関においてネットブート方式を利用する際には始業時といった端末を一斉に起動する際の負荷を想定し，この時にも端末を実用的な時間で起動するためにおよそ 20〜50 台での端末に対して 1 台のサーバが必要となっていた．この制限は規模のシステムにおいてサーバ台数が多くなることにつながり，管理における新たなコスト要因となっていた．

2.2 ネットブート環境における ReadCache 機構の利用

端末が起動する際にサーバから読み込むデータは，端末を何度起動しようともそれほど変化するものではなくキャッシュが効果的に働く．そこで、各端末のローカルディスク内にキャッシュ領域を作成し，サーバから読み込んだデータをそこに保持する “Read Cache” 機構が広く利用されるようになっている．端末を再起動してもキャッシュ内のデータは保持されるため，ディスクイメージが変化しない限りはキャッシュされているデータを再利用できるため，端末起動時にサーバに負荷をかけない．

“Read Cache” 機構におけるキャッシュ領域は，これまで次の 2 方式で確保された領域が利用されてきた．

2.2.1 専用のパーティションを作り，固定サイズのキャッシュ領域を作る方式

キャッシュデータを保存するための専用のパーティションを作り，そこにディスクイメージと同等のサイズを確保する．そしてディスクイメージの特定のオフセットからデータを読みこんだ際には，キャッシュ領域の先頭から同じオフセットの場所にキャッシュデータを保存する．

この方式は構造がシンプルであり，キャッシュ領域の開始オフセットを変えるだけで複数種類のディスクのキャッシュを行いやすい．しかし，ハードディスクは WriteCache など，

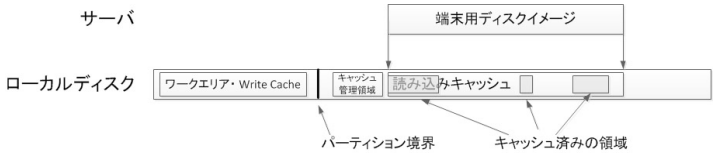

図 1 専用のパーティション内に読み込みキャッシュ領域を静的に確保する方式
Fig. 1 Static sized read cache storage in a dedicated partition.

ReadCache 以外の目的にも使われるため，端末からのアクセスが多発するとディスク上の複数の領域が頻繁にアクセスされることになり，ディスクアクセスに伴うヘッドシークが多発することになり速度低下する．また，ディスクイメージと同じサイズのキャッシュ領域が必要となるため，端末内には大きなサイズのディスクドライブが必要となりディスクに無駄が大きく，かつ導入コストがかかる．

2.2.2 OS の管理するファイルシステム内に動的サイズのキャッシュ領域を作る方式

ネットブートにより起動する OS が管理するファイルシステム内にキャッシュデータを貯めるためのキャッシュデータ用ファイルを作成する．ディスクイメージから読み込んだ順にキャッシュデータ用ファイルにデータを貯めていく．この場合，キャッシュのファイルのサイズは蓄積されたキャッシュの量となるため，ディスク領域を有効に活用できる．

図 2 ファイル内にキャッシュ領域を動的に確保する方式
Fig. 2 Dynamic sized read cache storage in a specific file.

この方式はディスクイメージ上のオフセットとキャッシュファイル上でのオフセットとを変換するテーブルが必要となる点で若干複雑なデータ構造とはなるが，ディスク上でのデータの配置をファイルシステムに任せてしまえる点で開発は容易となる．ただ，ディスク上でのデータの配置をファイルシステムに任せる弊害として，キャッシュファイル上でデータが連続していたとしても，それがディスク上で連続しているとは限らなくなる．

2.3 Solid State Drive （SSD） を利用した高速化

端末内のハードディスク速度の問題やヘッドシークの問題を解決するためにハードディスクを Solid State Drive（SSD） へと置き換える手法も検討できる．しかし，SSD は価格が高価であり置き換えが容易ではない点に加え，ハードディスクに比して容量を確保しづらいため，キャッシュ領域を静的に確保する形式では特に使いづらい．

3. 提 案 方 式

3.1 ReadCache とキャッシュデータの保存形式

本研究ではネットブート環境にける端末の起動時間の短縮を目標とするが，まずは多数の端末を一斉に起動した際においてもサーバに負荷をかけないように，端末の起動時間を安定化させるために ReadCache 機構の採用をする．

しかし，前章で述べたように従来の ReadCache 機構には「キャッシュ領域を静的に確保すると，ディスクに無駄が生じる.」「キャッシュ領域を独立したパーティションにすると，ヘッドシークが多発し起動に時間がかかる」等の課題がある．SSD を使うことでこれらの課題は解決できるものの，現状では SSD を各端末に備えるにはコスト面の課題が存在する．

そこで本研究では「従来型のハードディスク」上に構築されるキャッシュデータの形式を工夫することで，端末の起動時間の短縮を試みる．まず，ReadCache のキャッシュデータを「どの領域」に「どのような形式」で保存するべきかについて検討した．従来の ReadCache のデータの保存の方式を,「データをどこに保存するか」「データをどのような形式で保存するか」について以下のように分類する.（**表 1**）

表 1 キャッシュデータの保存領域と保存形式
Table 1 Recording area and recording format of cache data

		キャッシュを置く領域	
		OS 管理領域内	独立したパーティション
キャッシュ領域のサイズ	仮想ディスクと同じ（実装が容易）	X Type 1	○ Type 2 （**図 1**）
	キャッシュに貯めた量と同じ	○ Type 3 （**図 2**）	◎ Type 4 (**図 3**)

表 1 の Type 2 および Type 3 の特徴および課題については前章にて述べた．この研究ではキャッシュ領域のサイズを小さく抑えるために，キャッシュ容量がキャッシュに貯めた量となる「動的なキャッシュ領域」を用いる手法を採用する．そしてキャッシュデータの読み

込みに伴うヘッドシークを減らし起動時間を短縮するために，Type 4（図 3）の方式を採用する．この方式はキャッシュを動的に管理しつつ，ディスク内でのアロケーションも制御しなければならないため実装は複雑になる．

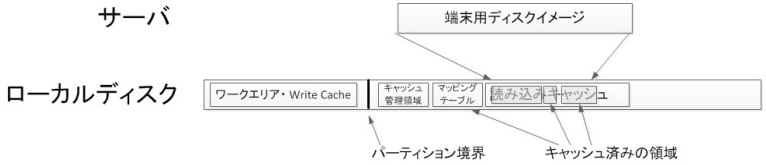

図 3 キャッシュ管理 Type 4 におけるディスク構成
Fig. 3 Data allocation on cache management type 4

3.2 キャッシュ領域の動的確保とマッピングテーブル

キャッシュデータは動的管理する機構とする．すなわちディスクイメージから読み込まれたデータをキャッシュ領域の先頭から順に保存していく方式となる．その際キャッシュデータはサーバからの読み込みに応じて追記されることになる．また、ディスクイメージの更新に伴いキャッシュ内容の更新が必要となるが、その場合には更新されたデータをキャッシュの末尾に追記して保存することにした．

この方式ではディスクイメージ上のオフセットとキャッシュ内のオフセットとの対応表を管理する必要がある．この対応表は，ディスクイメージをキャッシュの管理単位毎に分割した際のブロック数のエントリーが必要となる．各エントリーはキャッシュ領域内のオフセットを示すことになる．

3.3 プリフェッチ

まずキャッシュ領域を動的に確保する方式の ReadCache システムでキャッシュが空の状態から端末を起動すると，端末が起動する際にアクセスされるデータがキャッシュの先頭部分に蓄積されることに着目した．そしてキャッシュの領域を独立したパーティションに確保することで，端末が起動する際にアクセスされるデータがディスク上の一定の領域から連続して保持されることになる．そこで，端末の起動時にキャッシュ領域の先頭部分を一定サイズあらかじめ読み込んでおく．このアクセスはディスク上でも連続した領域を読み込むことになるため，高速に読み込める．このような処理により，端末起動時のアクセスのほとんどはメモリキャッシュにヒットすることが期待される．

図 4 キャッシュ領域のデータ構造
Fig. 4 Cache data structure.

4. 提案方式の実装

4.1 ネットブート機構と ReadCache の実装

ネットブート機構としては Citrix 社製 Citrix Provisioning Services （PVS）の Verion 5.6[1] を用いた．ネットブートされるクライアントとしては Windows 7 （64bit 版）を選択した．PVS はそれ自体では ReadCache の機能を持たないためシー・オー・コンヴ社製 ReadCache システム 3.6[2] を利用し，それをベースに拡張を行った．

4.2 マッピングテーブルの実装

キャッシュ専用のパーティションを確保し，そこに動的に管理されるキャッシュ領域を確保した．キャッシュデータは 1 セクタ単位で管理するようにした．マッピングテーブルは 1 セクタに対してセクタ位置を示す LBA（4 バイト）のテーブルとなる．(**図 4**)

実際のディスク上のブロック番号からキャッシュ上のブロック番号への対応表となる「Mapping Table」 を作成する．このマッピングテーブルとキャッシュとがディスク上に保存される．キャッシュをどこまで貯めたかを示すマーカー（EndMarker）をつくる．

キャッシュにないデータを読みこんだ際には以下の手順でキャッシュに蓄積する．(1) EndMarker を読み込んだサイズだけ後ろに移動する．(2) 読み込んだデータをキャッシュの末尾に保存する．(3) メモリ上の Mapping テーブルを更新する．(4) 更新された EndMarker をディスクに保存する．(5) 更新された Mapping テーブルをディスクに保存する．

この順序で処理をすることで，処理の途中で何らかの理由で突然停止をした場合でも，キャッシュの内容に不整合は生じない．

また Mapping テーブルおよびマーカーは，キャッシュへの蓄積の度に毎回ディスクに書き込むと，保存に時間がかかることになる．そこでこれらのテーブルのディスクへの書き込みは 1 分に 1 回程度の頻度に下げることにした．さらにディスクイメージよりも小さいキャッシュ領域でも動作するように，マーカーがキャッシュ領域の末尾に到達した際には端末は新たなキャッシュ動作をしないようにした．

4.3 プリフェッチの実装

端末起動時に読み込まれる量は，例として Windows7 であれば 300MB 程度である．このサイズであれば端末起動時に一時的にメモリを確保しても性能や挙動への影響はない．端末が起動し終えたタイミングを推測し，プリフェッチ領域を解放することもできる．

ところで，プリフェッチの読み込みには数秒かかる．(読み込みたいキャッシュサイズが 300MB 程度．ローカルディスクへのアクセスはおよそ 30-50MB/sec のため）端末の起動時にはサーバから読みこむためのデバイスとキャッシュディスクのデバイスの両方の初期化が同時に進む．プリフェッチの処理はキャッシュディスクの初期化の中で行うことになるが，プリフェッチによる読み込み処理に時間がかかるとその間にサーバ側からの読み込み処理が進み，キャッシュが実質的に機能しないという現象が発生した．この問題の対処のために，サーバから読み込むデバイスの初期化処理は，プリフェッチが完了するまで待つことにした．この対処によりプリフェッチされたデータを利用するようになったが，今度はサーバから読み込むデバイスの初期化に時間がかかりすぎると初期化処理に失敗したものとみなされることになる．この問題の対処のために，サーバ側のディスクの初期化処理は正常に終了させつつ，その後の読み込みリクエストに対してタイムアウトした旨の応答を返しリトライをさせるといった工夫が必要となった．

5. 評　　価

5.1 起動時間の評価について

従来の環境で端末を起動した際にはネットブートのための開始からログオンの完了までおよそ 101 秒かかっていたが，本提案の方式により 17 秒の短縮が出来た．(**表 2**）ログオンの完了は「ログオン処理が完了し，OS が利用可能な状態となるまでの時間」を計測した．(手作業で行っているため，数秒の誤差はある）

表 2　提案方式による起動時間 (秒)
Table 2　Startup time with proposed method(sec)

	従来方式	提案方式（プリフェッチなし）	提案方式（プリフェッチあり）
電源投入からログオン画面表示まで	36	23	25
電源投入から利用可能になるまで	101	84	84

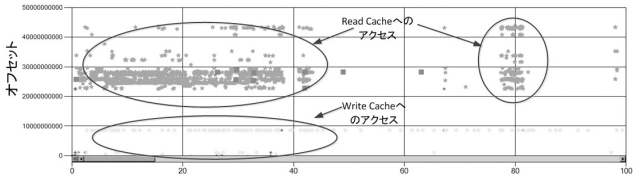

図 5　従来の ReadCache システムによるディスクアクセス
Fig. 5　Access pattern of the Local Disk with Conventional ReadCache System.

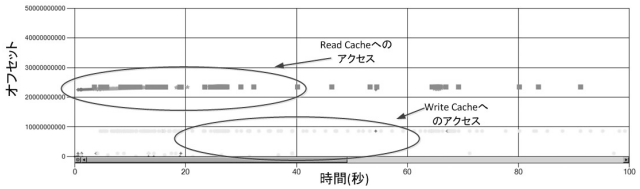

図 6　提案方式によりキャッシュ領域を動的管理した場合のディスクアクセス
Fig. 6　Access pattern of the Local Disk with proposed Read Cache System.

5.2 キャッシュ領域を動的管理にした効果について

従来用いていた CO-CONV 製 ReadCache システムはキャッシュ領域として専用パーティションを利用し，そこに静的なキャッシュ領域を構築していた．これを動的管理の機構に変更することで，ディスクへのアクセスパターンは次のように変化した．

キャッシュ領域へのアクセスは連続領域に対するアクセスに変化していて，ヘッドシークが発生しにくくなっていることがわかる．(**図 5 図 6**）

5.3 プリフェッチのサイズについて

次にプリフェッチ機能を有効にすることで，起動時のキャッシュ領域へのディスクアクセスがどのように変化するかを計測した．(**図 7 図 8**）プリフェッチ機構が有効な場合には起動

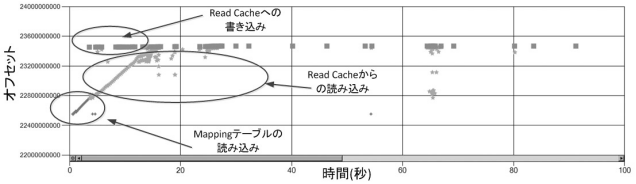

図 7 Prefetch 無効でのキャッシュ領域へのアクセス
Fig. 7 Access pattern to cache data w/o prefetch mechanism.

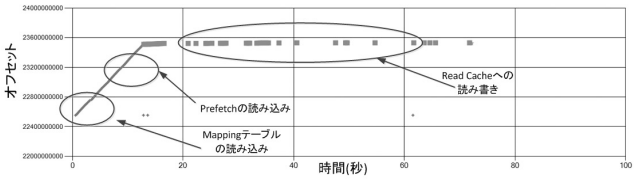

図 8 Prefetch 有効でのキャッシュ領域へのアクセス
Fig. 8 Access Pattern to cache data with prefetch mechanism.

時にキャッシュ領域をプリフェッチした後に，読み込み要求に対してプリフェッチされたメモリ上のデータから回答していることがわかる．今回の試験ではプリフェッチ領域（300MB）の読み込みに 5 秒弱の時間がかかっているが，その後のアクセスが高速になるためトータルでの起動時間はプリフェッチを利用しない場合と同程度になることが分かった。

プリフェッチのサイズは，OS の起動後ログオンするまでに読み込む量を目安としたいが，プリフェッチの読み込みを完了するまで起動処理を開始できないため，大きくすればよいというものではない．環境により大きく異なるが 300MB〜1GB 程度となると思われる．この点については,「適切なサイズを動的に決定する」「プリフェッチの読み込みを分割して順次読み込みつつ，起動処理を同期的に進めるという」といった対策を検討する必要がある．今回の計測ではプリフェッチによる起動時間の大幅な改善はできなかったが，これらの点を対策することで，今後さらなる改善をできるものと考えられる．

5.4 マッピングテーブルの保存作業のコスト

ディスクイメージを 40GB，キャッシュの管理単位を 1 セクタ（512 バイト），オフセットを 4 バイトで表現すると，マッピングテーブルのサイズは 320MB となる．このマッピングテーブルはキャッシュの内容を更新する度に一部分ないしは全体をキャッシュの管理領域に保存する必要があるため，保存に時間がかかることが問題になることが予測される．この問題にはキャッシュの管理単位を大きくすることで対処するのが妥当と思われる．

6. まとめと今後の課題

本研究ではネットブート環境における起動時間の短縮手法について検討した．

まず，ReadCache を利用したネットブート環境では端末内のハードディスクが ReadCache や WriteCache そしてワークエリアとしてさまざまな目的に使われるため，端末の起動時に多数のプログラムが一斉に動作した際にヘッドシークが多発することが起動に時間がかかる原因の一つとなっていることを計測により示した．この問題に対処するために，ReadCache のキャッシュを保存する際の保存先とデータ構造について検討し，キャッシュデータは専用のパーティションに動的な構造で確保することでヘッドシークが少なくなることを示した．

次に端末の起動時に読み込まれるデータは端末を何回起動しようともほぼ同じデータがほぼ同じ順序で読み込まれると推測されることから，キャッシュのデータをあらかじめメモリに読み込んでおくプリフェッチ機能について提案をした．そして提案方式の実装を行い，キャッシュ構造を変更したことでプリフェッチを容易に行え，かつプリフェッチによる起動時間の短縮の可能性について調査した．

なお提案方式を評価した結果として，キャッシュ構造を動的にしたことにより必要となったマッピングテーブルの読み込みや書き戻しにかかる時間が大きな割合を占めていることが分かった．この点を改善するにはキャッシュの管理サイズを適切に大きくすることが必要と思われる．また，プリフェッチ機能は起動時におけるディスクへのランダムアクセスを避けるために有効であることは確認できたが，サイズを大きくするとその分プリフェッチの読み込みに時間がかかることになる．このサイズをどの程度とすべきかは今後の検討課題となる．

最後に，今回の研究においては提案方式の評価を試験的に作られた環境において行ったが，今後実環境においても同様に評価を行い，その際に最適なパラメータを見つけ出すことも課題となる．

参 考 文 献

1) Xen Desktop (Provisioning Services), http://www.citrix.com/English/ps2/products/product.asp?contentID=163057&ntref=prod_cat
2) ReadCache システム 3.6, http://www.co-conv.jp/product/readcache/