そろそろキャッシュについて勉強しなおそうか2017夏

JCFUG 2017.07.28

そろそろキャッシュについて勉強しなおそうか2017夏

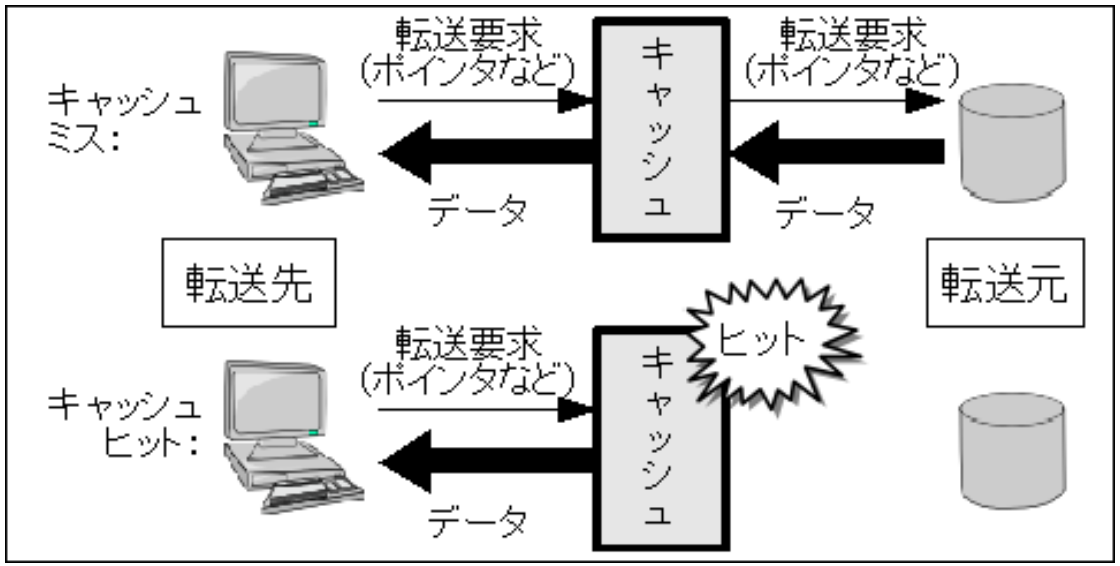
* 今回の内容
  * キャッシュについて（一般論）
  * なぜ今（更）キャッシュなのか
  * CFにおけるキャッシュ機能概説
  * 具体的な使用方法など

キャッシュ(コンピュータシステム)

* キャッシュ (英: cache) は、CPUのバスやネットワークなど様々な情報伝達経路において、ある領域から他の領域へ情報を転送する際、その転送遅延を極力隠蔽し転送効率を向上するために考案された記憶階層の実現手段である。実装するシステムに応じてハードウェア・ソフトウェア双方の形態がある
* 転送元と転送先の中間に位置し、データ内容の一部とその参照を保持する。データ転送元への転送要求があり、それへの参照が既にキャッシュに格納されていた場合は、元データからの転送は行わずキャッシュが転送を代行する
  * https://ja.wikipedia.org/wiki/キャッシュ (コンピュータシステム)

キャッシュ(コンピュータシステム)

* キャッシュが転送を代行した状態をキャッシュヒット、キャッシュに所望のデータが存在せず元データから転送する状態をキャッシュミスという
  * https://ja.wikipedia.org/wiki/キャッシュ (コンピュータシステム)



Web開発で使う一般的なキャッシュシステムの種類

* memcached
  * 汎用の分散キャッシュサーバー（独自プロトコルで通信）
  * メモリキャッシュのみ
  * Amazon ElastiCacheのエンジンとして選択可能
* Redis
  * NoSQL DBだが、キャッシュの保存先として使うことがある
  * ディスクへの書き出しも可能（永続化）
  * Amazon ElastiCacheのエンジンとして選択可能
* Ehcache
  * Java製の分散キャッシュシステム
  * ディスクへの書き出しも可能（永続化）
  * レプリケート機能やサーバ機能（REST / SOAP）などもあるが、どちらかというとキャッシュライブラリ的な使われ方で、やはりJavaからの利用がメイン？
  * 現在、CFの組み込みキャッシュ機構として採用されている（CF9+）

なぜ今キャッシュなのか

* Ehcacheを搭載したColdFusion9がリリースされたのが2009年後半
* memcachedが検索トレンドピークになっていたのも2009～2010年あたり（Googleトレンド調べ）
* サイトの大規模化、スマホ普及などによるトラフィックの増大により、Web界隈でキャッシュの存在感が増した？

* それから8年・・・

なぜ今キャッシュなのか

* CF8-
  * キャッシュ機能のメインがページ全体のコンテンツキャッシュだったので何となく使いにくい印象を与えていた
  * オブジェクト等のキャッシュは自前でApplicationスコープなどに入れて管理する必要があった
  * クエリーは昔からキャッシュできていた
* CF9+
  * ページ全体以外にもキャッシュオプションが増えた
  * オブジェクトのキャッシュなども組み込み機能で可能になった
  * Ehcache導入

  * で、具体的には今ってどうなってるのでしたっけ？

CFにおけるキャッシュ機能

* 大まかに分けて三種類のキャッシュ機能がある
  * ページキャッシュ
    * 出力コンテンツそのもののキャッシュ
    * ページの部分キャッシュ（CF9+）
  * オブジェクトキャッシュ（CF9+）
    * その他CF変数のキャッシュ
    * Ehcacheそのものの機能
  * クエリーキャッシュ
    * クエリー結果のキャッシュ

CFにおけるキャッシュ機能

* <cfcache>
  * ページコンテンツのキャッシュ（クライアントサイド/サーバサイド）
  * オブジェクトキャッシュ
* CacheGet / CachePut / CacheRemove
  * オブジェクトキャッシュ
* <cfquery> / queryExecute
  * クエリキャッシュ

* どれも内部的にはEhcacheを使っている
  * Ehcache上ではページ、オブジェクト、クエリで、異なる3つのデフォルト設定というような扱い
  * クエリキャッシュのみ、CFIDEやApplication.cfcの設定で旧キャッシュ機構を使うようにすることも可能（特に戻す必要性はなさそうだが？）
    * http://help.adobe.com/ja_JP/ColdFusion/10.0/Developing/WSe61e35da8d3185187f5cb36b135869d3836-7ffe.html
    * 恐らく元々は機能別にバラバラに実装されていた？

<cfcache>

* コンテンツの部分キャッシュ

```
<cfset i = 1>
<cfcache action="serverCache">
        この中がキャッシュされる
        #Now()#

        <cfset i++> <!--- 二度目は実行されない --->
</cfcache>
<cfset j = i * 2>    <!--- キャッシュ時は結果が変わる--->
```

* キャッシュされた部分は処理自体がスキップされ、保存されたテキスト内容に差し替えられるような動作イメージ
* バグの温床になりそうなので、キャッシュ範囲内になるべく処理を書かない方が良さそう

<cfcache>

* ページ全体のキャッシュ

```
<cfcache action="serverCache">

<!--- これは空の部分キャッシュになるため注意 --->
<cfcache action="serverCache" />
```

* キャッシュが有効な場合はページ全体を処理しない

<cfcache>

* ページ全体のキャッシュ

```
<cfcache action="serverCache">

<!--- これは空の部分キャッシュになるため注意 --->
<cfcache action="serverCache" />
```

* ~~キャッシュが有効な場合はページ全体を処理しない~~
  絶対そうとは言い切れない模様

<cfcache>

* ページ全体のキャッシュ

```
<cfset session.count++>
<cfcache action="serverCache">

<cfdump var="#session#">
```

* 上記のような処理を複数回実行するとキャッシュヒットした場合でも、内部的にはカウントアップされている（CF10で確認）

<cfcache>

* ページ全体のキャッシュ

```
<cfcache action="serverCache">
<cfset session.count++>

<cfdump var="#session#">
```

* この場合はカウントアップされない

<cfcache>

* ページ全体のキャッシュ

```
<cfset session.count++>
<cfcache action="serverCache">
<cfset session.count++>

<cfdump var="#session#">
```

* キャッシュヒットした場合でも<cfcache>記述箇所以前の処理は実行される、っぽい
* コンテンツ自体は全体がキャッシュされており、<cfcache>記述以前の出力内容がキャッシュされないわけではない
* 事情は分からなくもないがちょっと意表をついているので、ページ全体をキャッシュする場合はなるべく上の方に書いた方が無難そう

<cfcache>

* クライアンドサイドのキャッシュ（ページ全体）
  <cfcache action="clientCache">

* クライアントキャッシュ = Cache-Controlヘッダの操作
  * 常に期限切れにして更新確認はさせるようなヘッダ構成なので、キャッシュ設定を変えた場合なども変更は反映される（リクエスト自体を飛ばさせないような設定ではない）

    Expires: Mon, 05 Jun 2017 07:31:34 GMT　←処理した時間（常に期限切れ）
    Cache-Control: public,max-age=-1　←常に期限切れ

    * 更新があった場合→200を返す
    * 更新がない場合→304を返す（コンテンツボディは転送しない）

* 部分キャッシュ時も設定可能だが、その場合も応答ヘッダにキャッシュコントロールを入れてしまうため期待した動きにはならない（全体がキャッシュされたような動きになる）

<cfcache>

* オブジェクトキャッシュ

```
<cfset id = 1>

<!--- getでキャッシュから取得 --->
<cfcache action="get" name="obj" id="cached_obj_#id#">

<!--- キャッシュにない場合は変数が未定義のままになる --->
<cfif IsDefined("obj")>
        <cfdump var="#obj#">

        <!--- キャッシュから削除 --->
        <cfcache action="flush" id="cached_obj_#id#">
<cfelse>
        obj is undefined.

        <cfset obj = { "test" = 1 }>

        <!--- putでキャッシュに追加。value属性には#が必要 --->
        <cfcache action="put" value="#obj#" id="cached_obj_#id#">
</cfif>
```

* オブジェクトキャッシュの場合id属性は必須で、キャッシュ内容を識別するために使う

CacheGet / CachePut / CacheRemove

* オブジェクトキャッシュ

```
<cfset id = 1>

<!--- getでキャッシュから取得 --->
<cfset obj = CacheGet("cached_obj_#id#")>

<!---キャッシュにない場合は変数が未定義のままになる --->
<cfif IsDefined("obj")>
        <cfdump var="#obj#">

        <!--- キャッシュから削除 --->
        <cfset CacheRemove( "cached_obj_#id#")>
<cfelse>
        obj is undefined.

        <cfset obj = { "test" = 1 }>

        <!--- putでキャッシュに追加 --->
        <cfset CachePut("cached_obj_#id#", obj)>
</cfif>
```

* <cfcache>と動きは同じ。

<cfquery> / queryExecute

* クエリーキャッシュ
  * 同じ SQL 文（パラメータ含む）、データソース、クエリー名、ユーザー名、およびパスワードを使っている場合にキャッシュが使われる
  * キャッシュ期限の設定が必須で、以下のいずれかを指定することでキャッシュが有効になる

  * <cfquery cachedAfter="#CreateDateTime(2017, 7, 28, 15, 0, 0)#">
    * 指定した時刻以降に生成されたキャッシュがある場合、キャッシュを使用する
      * 例：2017/07/28 15:00:00以降の最初のアクセスで生成された結果がその後ずっとキャッシュされる
        （実際には日付をNow() + Xのように変えながら使用？）
      * なぜexpireの期限ではなく、開始時間を指定するのか・・・？
        * 日次バッチ等で特定時間以降にキャッシュを適用するといった用途を想定しているとのこと
          * 例えば銀行などで15時以降は窓口を閉めて固定化するためキャッシュを使う、などといったイメージ
          * 日付部分だけ可変にし、時刻を固定にして使う
        * また、例えば10分単位でキャッシュしたいが、毎時0分でバッチ更新があるためその時だけは最長10分待たずに即時反映したい、といったような場合に、現在時刻+10分刻み（切り捨て）で時刻指定を調整することで、細かいコントロールが可能となる
  * <cfquery cachedWithin="#CreateTimeSpan(0, 0, 5, 0)#">
    * キャッシュ生成から指定した期間の間キャッシュを使用する
      * 例：5分間キャッシュされる。5分経過後はキャッシュは使われないが、経過後の最初のアクセスでまたキャッシュされ、再度5分間キャッシュ状態が続く
        （5分以内にアクセスがあってキャッシュが使われても延長はされない）

<cfquery> / queryExecute

* クエリーキャッシュ
  * <cfset qArt = QueryExecute(
    "select * from art where ARTID < :artid and artistid=:aid "
    , {artid={value=100}, aid=2}
    , {
    datasource="cfartgallery"
    , cachedAfter=CreateDateTime(2017, 7, 28, 15, 0, 0)
    }
)>
  * <cfset qArt = QueryExecute(
    "select * from art where ARTID < :artid and artistid=:aid "
    , {artid={value=100}, aid=2}
    , {
    datasource="cfartgallery"
    , cachedWithin=CreateTimeSpan(0, 0, 5, 0)
    }
)>
  * 三つ目のクエリーオプションに<cfquery>と同じ属性名が渡せる

キャッシュの破棄

* ページキャッシュとオブジェクトキャッシュにもクエリーキャッシュの様な破棄タイミングの指定オプションがある
  * idleTime
    * キャッシュへのアクセスが一定期間発生しない場合に破棄
    * <cfcache> / CachePut
  * timeSpan
    * キャッシュされてから一定期間経過した場合に破棄
    * <cfcache> / CachePut
  * ID指定で明示的に破棄
    * <cfcache action="flush"> / CacheRemove
      * <cfquery> / queryExecuteで生成したクエリキャッシュにもIDは付けられる

キャッシュの破棄

* dependsOn
  * 指定した変数が変化した場合に破棄
  * <cfcache> ※ページキャッシュのみ
* expireURL
  * 指定された URL またはパターンに一致するページキャッシュを破棄
  * <cfcache action="flush"> ※ページキャッシュのみ
* usequerystring
  * クエリ文字列（URL変数）別にキャッシュ生成
    * かつ、URL変数がdependsOnの対象に勝手に入り、変化した場合にページキャッシュを破棄。結局上記とほぼ同じことではあるが、URLについていなくてもURLスコープをいじると影響がある
  * <cfcache> ※ページキャッシュのみ
* cachedWithin="0"で破棄
  * <cfquery> / queryExecute
  * 0 = CreateTimeSpan(0, 0, 0, 0)と同義
  * CF8-で能動的に破棄する手段がないため使われていたTips的なテクニック

キャッシュの破棄

* idleTime
  * キャッシュへのアクセスが一定期間発生しない場合に破棄
  * CreateTimeSpanか「小数で表した日数」で指定

```
<!--- 6時間--->
<cfcache idleTime="#CreateTimeSpan(0, 6, 0, 0)#">

<!--- 6時間 = 1/4日 = 0.25 --->
<cfcache idleTime="0.25">
```

  * cachedWithinとは異なり、時間内にアクセスがあった場合は期間が延びる（カウントがリセットされる）
    * cachedWithinは指定の期間が過ぎたら強制破棄

キャッシュの破棄

* timeSpan
  * キャッシュされてから一定期間経過した場合に破棄
  * CreateTimeSpanか「小数で表した日数」で指定

```
<!--- 6時間--->
<cfcache timeSpan="#CreateTimeSpan(0, 6, 0, 0)#">

<!--- 6時間 = 1/4日 = 0.25 --->
<cfcache timeSpan ="0.25">
```

  * cachedWithinと同じ動作

キャッシュの破棄

* ID指定で破棄

```
<!--- キャッシュから削除 --->
<cfcache action="flush" id="cached_obj_#id#">

<!--- キャッシュから削除 --->
<cfset CacheRemove("cached_obj_#i#")>

<!---
        regionを指定してクエリキャッシュを削除（CF10+）
        ※cfquery実行時にcacheID属性を使わないとオブジェクトのリファレンスが入っており、CFから指定できない模様
--->
<cfquery ～ cacheID="cached_obj_#i#">
<cfset CacheRemove("cached_obj_#i#", true, "query", true)>

<!---
        regionを指定してテンプレートキャッシュを削除（CF10+）
        ※IDは独自ロジックで自動的に決まっているが、CacheGetAllIds("template")で一覧を取得可能
          また、ページキャッシュ削除用にはexpireURLなど他のオプションもある
--->
<cfset
CacheRemove("HTTP://192.168.0.10/JCFUG/PAGE_CACHE.CFM_PAGEID:F1C2501EBB45A4D8F65FD1
667035D0B0_LINE:6", true, "template", true)>
```

キャッシュの破棄

* dependsOn
  * 対象はページキャッシュのみ
  * 変数名リストに指定したいずれかの変数の内容が変わった際にキャッシュを破棄、更新する

```
<cfparam name="Form.page" default="1">
<cfparam name="Form.num" default="20">

<!---変数名のカンマ区切りのリストを指定 --->
<cfcache action="servercache" dependsOn="Form.page,Form.num">
```

キャッシュの破棄

* expireURL
  * 対象はページキャッシュのみ
  * ページキャッシュはURL毎に保存されている
  * 指定したURLパターンにマッチしたキャッシュを全て削除する
  * 「*」でワイルドカードが使用可能。前方一致

```
<!---
    ワイルドカードは複数指定可能な模様
    末尾にワイルドカードが無くても前方一致でマッチする
        例：/cfcache_page.cfm?a=1などにもマッチする
--->
<cfcache action="flush" expireURL="*/jcfug/*/cfcache_page.cfm">
```

キャッシュの破棄

* usequerystring
  * 対象はページキャッシュのみ
  * クエリ文字列（URL変数）別にキャッシュ生成する
  * URL変数が自動的にdependOn属性に入った扱いになる

```
<!---
    クエリ文字列に含んでいなくとも、URLスコープの変数が
    変化した場合、キャッシュが再生成される
--->
<cfparam name="Url.page" default="1">
<cfparam name="Url.num" default="20">

<cfcache action="serverCache" usequerystring="true">
```

統計情報の取得

* キャッシュがどの程度使われているかの情報を取得できる
  * CacheGetMetadata("cache_id")
    * 個別キャッシュ単位での統計情報の取得

struct	
cache_hitcount	23
cache_misscount	12
createdtime	{ts '2017-06-08 13:22:52'}
hitcount	0
idletime	86400
lasthit	{ts '2017-06-08 13:22:52'}
lastupdated	{ts '2017-06-08 13:22:52'}
name	OBJECT
size	404
timespan	86400

統計情報の取得

* CacheGetSession("template").getStatistics()
  * キャッシュ種類(region)別の統計情報オブジェクトの取得
    * template（ページキャッシュ）
    * object
    * query
    * その他ユーザー定義したもの
  * net.sf.ehcache.Statisticsを返却

getAssociatedCacheName()	TEMPLATE
getAverageGetTime()	0.138386
getAverageSearchTime()	0
getCacheHits()	298
getCacheMisses()	53
getDiskStoreObjectCount()	0
getEvictionCount()	0
getInMemoryHits()	298
getInMemoryMisses()	50
getMemoryStoreObjectCount()	3
getObjectCount()	3
getOffHeapHits()	0
getOffHeapMisses()	0
getOffHeapStoreObjectCount()	0
getOnDiskHits()	0
getOnDiskMisses()	0
getStatisticsAccuracyDescription()	Best Effort

Ehcacheの設定

* CacheGetProperties
  * キャッシュ設定の取得
    * ディスクに書き出すかどうか、キャッシュする最大数など
* CacheSetProperties
  * キャッシュ設定の変更
* Application.cfc - this.cache.configfile
  * ehcache.xml設定ファイルの指定
    * http://help.adobe.com/ja_JP/ColdFusion/10.0/Developing/WSe61e35da8d3185187f5cb36b135869d3836-7fff.html
  * より細かい設定が可能
  * 独自にregion(ehcache.xmlではcacheNameに相当)も定義可能

Ehcacheへの直接アクセス

* CacheGetSession("region名")
  * object of net.sf.ehcache.Cacheを返却
* CreateObject('java', 'net.sf.ehcache.CacheManager').getInstance()
  * Ehcacheのメソッド等を直接呼び出せる
  * ehcache 2.x系（3.xはパッケージ名が異なる）
    * http://www.ehcache.org/apidocs/2.10.4/index.html

Ehcacheへの直接アクセス

* <cfset cacheManager = createObject('java', 'net.sf.ehcache.CacheManager').getInstance()>
* <cfset regions = cacheManager.getCacheNames()>
* <cfdump var="#regions#">

array	
1	TEMPLATE
2	QUERY
3	customcache
4	cfcache_fragmentTEMPLATE
5	OBJECT

regionについて（CF10+）

* region = cacheName
  * template, object, queryなど組み込みのregionは定義済みになっており、CFのキャッシュ関数はそれぞれ特定の領域を使用する
    * 厳密にはobject以外は最初に使用したときに生成される模様
    * ページキャッシュ = template
    * オブジェクトキャッシュ = object
    * クエリーキャッシュ = query
* CF10+では各キャッシュ機能でregionを指定することができ、これによりehcache.xmlで設定したユーザー定義のregion(cacheName)も利用可能となる

regionについて（CF10+）

* 一部機能のインターフェース（あるいはマニュアル）に
やや混乱が見られる？
  * CF9当時、アップデータによって機能追加された関係でバラツキが生じていた、らしい
  * <cfcache>
    * key属性(CF9)とregion属性(CF10+)は同じことなのか？
    * 元々ユーザー定義領域を使うため、<cfcache>にkey属性があったが、CF10で汎用的にregion属性が追加されている
    * サンプルではkey属性を使っているが、実際にはregionに統合？
  * CacheGetProperties
    * typeパラメータ(CF9)とregionパラメータ(CF10+)も役割的には同じっぽいが二つ書かれている
    * こちらは英語版ではregionに統一されていた

大規模環境向け拡張設定

* その他、Ehcacheのレプリケーション設定、Terracotta連携による分散オブジェクトキャッシュ、BigMemory Goによるオフヒープメモリの利用など、Ehcache向けの連携機能は一通り設定できる模様
  * Setting up ColdFusion 10 to replicate ehCache
    * https://blog.pengoworks.com/index.cfm/2014/9/24/Setting-up-ColdFusion-10-to-replicate-ehCache
  * Scale ColdFusion with Terracotta Distributed Caching for Ehchache
    * https://www.slideshare.net/ColdFusionConference/scale-coldfusion-with-terracotta-distributed-caching-for-ehchache
  * Using Coldfusion and BigMemory Go
    * http://terracotta.org/documentation/4.1/bigmemorygo/integrations/coldfusion

そろそろキャッシュについて勉強しなおそうか2017夏

ご清聴ありがとうございました