

All about CFThread

Rupesh Kumar
Computer Scientist
Adobe

June 21, 2008
<http://coldfused.blogspot.com>

Agenda

- Introduction
- Syntax
- Internals
- Things to be careful about
- Administration & monitoring
- Q&A

Introduction

- Multi-Threading allows multiple tasks to run in parallel.
- Harnesses the processing power of the processor to give a huge performance improvement.
- ColdFusion and all the servers are multi threaded
 - All the requests run in parallel in separate threads
- ColdFusion 8 takes it to cfml using CFThread.

Multi-Threading in CFML – CFThread

- Allows cf page to launch multiple tasks to run in parallel.
- New threads run asynchronous to the page
- Use cases
 - Fire and Forget
 - Fire and wait to finish

Use Case I – Fire and Forget

- Lot of tasks for which end user need not be made to wait
- Image web application that imports images from user's flickr account, resizes it and adds to user's account.
- Blog application that imports blog entries from user's other account.


Use case II – Fire and wait to finish

- Lot of independent tasks that need not be synchronous
- Travel portal that needs to get quote from different airlines and hotels
- RSS aggregator


First Look

```
<cfthread name="mythread">  
  <!--- any cfml code --->  
  <cfset avail = getSeatAvail(airline)>  
  <cfset price = getPrice(airline)>  
</cfthread>
```

Execution Flow




```
<cfset = 10>  
<cfset threadname = "T1">  
<cfthread name="#threadname#" myurl="http://cfunited.com/">
```



```
<cfhttp url="#myurl#"  
        method="GET" result=#cfunited">  
<cfmail attributeCollection=attrs>  
    #cfunited.FileContent#  
</cfmail>  
<cfset thread.result = #cfunited#>
```

```
</cfthread>  
<cfset foo()>  
...  
...
```

```
<cfthread action="join" />  
<cfdump var="#T1.result#">
```

 Request Thread

 CFThread

How to create a thread?

```
<cfthread name="t" [action="run"] [priority="high"] >  
  <!-- any cfml code --->  
</cfthread>
```

- Name

- Thread name must be unique in the request
- Two different requests can have the same thread name.

- Priority

- Normal (default), high, low
- Same as JVM thread priority.
- Request thread is at normal priority
- In case of long running background thread (fire & forget), you can set a lower priority.

Internals

- Thread body is extracted and a function is created dynamically.
- This function is invoked in a new thread that runs parallel to the request.
- This is why you see a new UDF when you dump the variable scope.

- example

```
<cfthread name="t">  
  <cfoutput>say hi from thread </cfoutput>  
</cfthread>  
<cfdump var="#Variables#">
```

- CFThread != Java/Native Thread

Key Concepts

- Can live much beyond the request thread
- Since it is internally a function, it has its own local scope.
- You can define a variable as var.
- Any variable declared without any prefix also goes in this local scope.
- Can access all the scopes and its variables
 - You can not write to the session, cookie scope after the request is done

Example : Fire n Forget

```
<cfset importUrl = Form.importUrl>
<cfset dirToImport = getUserDirectory(>

<cfthread name="importThread">
    <cfset downloadImages(importUrl, dirToImport)>
    <cfset resizeImages(dirToImport)>
    <cfset addImages(dirToImport)>
    <cfset sendMail(>
</cfthread>

<cfoutput>Thanks for using Pixel! The import is in
progress and we would notify you once the import
completes. </cfoutput>
```

Wait for thread to Finish

- Action "Join"
- `<cfthread action="join" name="t" />`
- `<cfthread action="join" name="t1, t2, t3" />`
- `<cfthread action="join" />`
- `<cfthread action="join" name="t1,t2" timeout=5000>`

- Any thread can join with any cfthread. Waiting thread can be request thread also.

Example : Fire n wait

```
<cfthread name="adbe">  
  <cfset Variables.q_adbe= GetStockQuote( "ADBE" )>  
</cfthread>
```

```
<cfthread name="msft">  
  <cfset Variables.q_msft=GetStockQuote( "MSFT" )>  
</cfthread>
```

```
<cfthread name="apl">  
  <cfset Variables.q_apl=GetStockQuote( "AAPL" )>  
</cfthread>
```

```
<cfthread action="join" name="adbe,msft,apl">  
<cfoutput>adbe : #q_adbe.price#<br>  
          msft : #q_msft.price#<br>  
          apl : #q_apl.price#<br></cfoutput>
```

Puzzle 1

- Copy a set of files from one directory to another using array

```
<cfset files = ["file1.txt","file2.txt","file3.txt",  
               "file4.txt","file5.txt"]>
```

```
<cfset src="c:/tmp/1">
```

```
<cfset dest="c:/tmp/2">
```

```
<cfloop from=1 index="i" to=#ArrayLen(files)#>
```

```
  <cfoutput>Copying #src#/#files[i]#</cfoutput><br>
```

```
  <cfthread name="thread#i#">
```

```
    <cffile action="copy" source="#src#/#files[i]#"  
           destination="#dest#">
```

```
  </cfthread>
```

```
</cfloop>
```

Puzzle 2

- Copy a set of files from one directory to another using query

```
<cfset src="c:/tmp/1">
```

```
<cfset dest="c:/tmp/2">
```

```
<cfdirectory action="list" directory="c:/tmp/1"  
             listinfo="name" name="files" />
```

```
<cfset i = 1>
```

```
<cfloop query=files>
```

```
  <cfoutput>Copying #src#/#files.name#</cfoutput><br>
```

```
  <cfthread name="thread#i++#">
```

```
    <cffile action="copy" source="#src#/#files.name#"  
            destination="#dest#">
```

```
  </cfthread>
```

```
</cfloop>
```

Thread safety

- Both the above examples are thread unsafe.
- Pass the variables as cfthread attributes.
- Can pass any attribute directly or in attributecollection.
- Access those attributes as attributes.xxx

```
<cfthread name="thread#i#" filename="#src#/#files[i]#">  
  <cffile action="COPY"  
    source="#src#\#attributes.filename#"   
    destination="#dest#">  
</cfthread>
```

- Can also be accessed directly but prefer to use attributes.xxx
- Attributes are duplicated (deep copy) to maintain thread safety. (WHY??)

Handling Output

```
<cfloop from=1 to="5" index="i">
  <cfthread name="t#i#" no='i'>
    This is thread 't'<cfoutput>#no#</cfoutput>
  </cfthread>
</cfloop>
```

- Where did the output go?
- For thread safety and consistency of output, cfoutput inside thread body will not write to the page response.
- Each thread maintains its own output buffer to which all the output are written.
- It can be retrieved using "OUTPUT" key of thread scope.
- Use cfdump with output=console to debug easily.

Handling Error

- Error in CFThread does not terminate the request and error does not go to the browser.
- Error handling is done like normal error handling in cf page.
- Use cftry/cfcatch or try/catch
- If error is not caught, the error will be added to "ERROR" meta-data of the thread scope.
- The error caught is also logged to application.log and exception.log

Thread Scope

- Shared scope between threads.
- Used to make thread specific data available to other threads.
- Only owner thread can write but anyone can read.
- Accessed using thread name.
- Can also be accessed using "Thread" by owner thread.
 - You can use this to find if you are inside a thread.
- A subscope of CFThread scope
- Available to threads in the same request. Other request can not see this.
- Also contains thread metadata

CFThread scope

- A new scope
- Contains all the thread scopes spawned in the request
- Lasts as long as the requests or until the last thread of request runs
- Can be accessed using 'cfthread'.
- When do I use it? - dynamic thread name

```
<cfset threadname="mythread">
```

```
<cfset myoutput = cfthread[threadname].output>
```

Thread Meta-data

- Name
- Priority
- StartTime
- ElapsedTime
- Output
- Error
- Status - NOT_STARTED | RUNNING | WAITING | COMPLETED | TERMINATED

```
<cfthread name="t1">  
  Hello from thread!  
</cfthread>
```

```
<cfdump var="#t1#">  
<cfthread action="join" />  
<cfdump var="#t1#">
```

t1 before join - struct	
ELAPSEDTIME	0
NAME	T1
OUTPUT	[empty string]
PRIORITY	NORMAL
STARTTIME	{ts '2008-04-14 11:46:38'}
STATUS	RUNNING

t1 after join - struct	
ELAPSEDTIME	1766
NAME	T1
OUTPUT	Hello from thread!
PRIORITY	NORMAL
STARTTIME	{ts '2008-04-14 11:46:38'}
STATUS	COMPLETED

Other cfthread actions

▪ Sleep

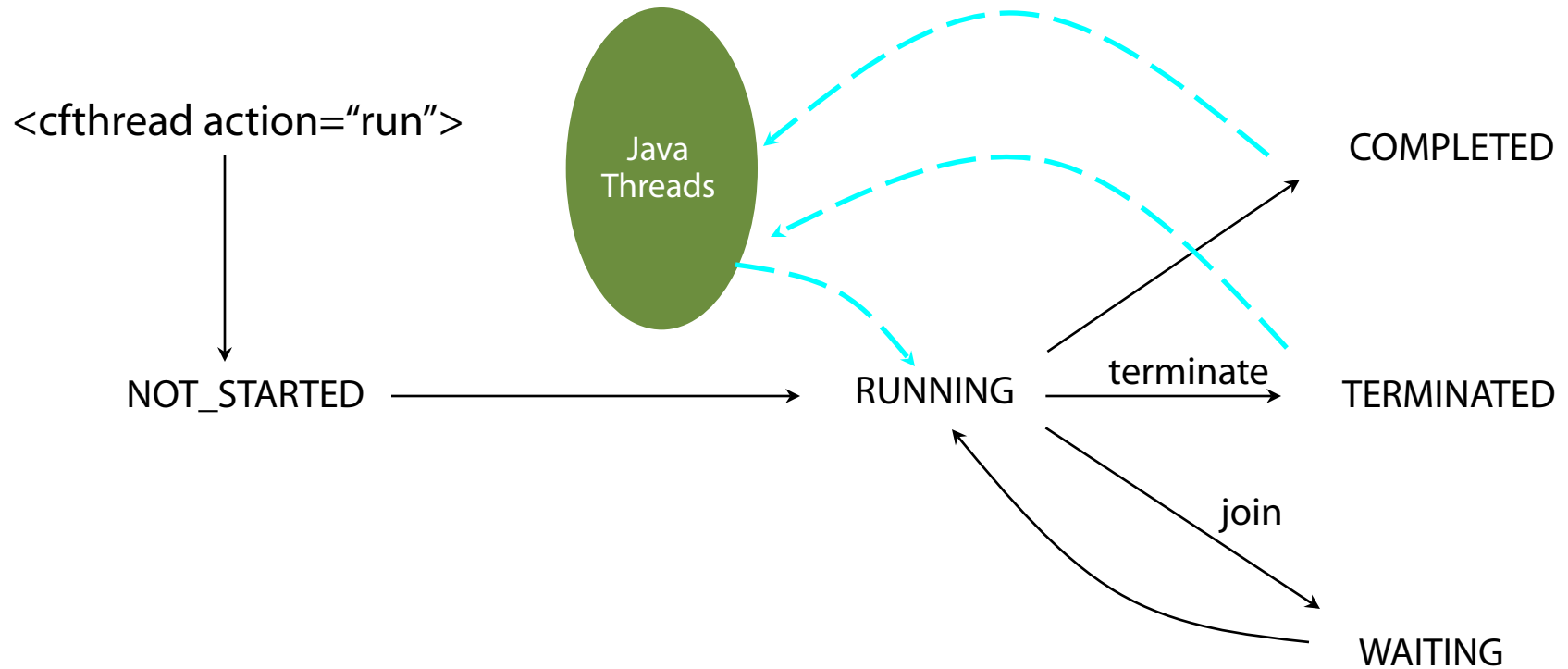
- Suspends the **current** thread for given duration.
- `<cfthread action="sleep" duration="5000" />`
- Inbuilt function **sleep**(5000)
- Does not free up the java thread.

▪ Terminate

- Terminates a thread.
- Should not be abused as it can lead to inconsistency.

```
<cfthread action="terminate" name="t1,t2" />
```

Thread Life Cycle



Working with shared resource

- For thread safety, lock the resource if accessed by multiple threads
- Use `cflock`
- Lock on request object if accessing request scope or any subscope (form, url etc)
- `cflock` now supports request scope

Deadlock

- Care needs to be taken to prevent deadlock.
- Deadlock scenarios
 - With join.
 - T1 joins with t2 and T2 joins with t1.
 - With lock
 - Thread 1 locks resource 1 and tries to lock resource 2
 - Thread 2 locks resource 2 and tries to lock resource 1.

Administration

- "Maximum number of java threads available for CFTHREAD" : 10 (default).
- There is a pool of java threads which takes care of executing cfthreads.
- If no java thread is free, cfthread will be queued until a java thread is free to execute it.
- If java threads are idle, they time out after 5 mins.
- The number of threads should not be very high else it will cause lot of context switching affecting the server performance.

Monitoring

- See all running CFthreads
 - threadname, page spawned from, elapsed time.
- Kill non-responsive or long running cfthread
- Slowest cfthreads
 - Time taken by slow tags and function in the thread
- cfthreads by memory
- Alerts for cfthread
- Admin apis available for all.

Monitoring

CF ADOBE® COLD FUSION® MONITOR STOP MONITORING START PROFILING START MEMORY TRACKING LOGOUT

OVERVIEW STATISTICS ALERTS SNAPSHOTS Current Server time : Monday, Apr 14, 2008 11:51:56 AM

REQUEST STATISTICS > Active ColdFusion Threads

Lists all active ColdFusion threads on the server that are launched by the cfthread tag.

ColdFusion Thread Name	Spawned Template Path	Time Taken (seconds)	Java Thread Name
T1	C:\Work\...\cfthread\junk.cfm :: 2	52.846	cfthread-0

Left sidebar menu: REQUEST STATISTICS (Active Requests, Active ColdFusion Threads, Slowest Requests, Slowest ColdFusion Threads, Active Sessions, Cumulative Server Usage, Highest Hit Counts, Template Cache Status, Request Throttle Data), MEMORY USAGE, DATABASE, ERRORS.

CF ADOBE® COLD FUSION® MONITOR STOP MONITORING START PROFILING START MEMORY TRACKING LOGOUT

OVERVIEW STATISTICS ALERTS SNAPSHOTS Current Server time : Monday, Apr 14, 2008 11:47:54 AM

REQUEST STATISTICS > Slowest ColdFusion Threads

Lists the slowest ColdFusion threads on the server that are launched by the cfthread tag.

List up to 20 threads slower than 1 seconds

Spawned Template Path	Avg Response Time (seconds)
C:\Work\...\cfthread\barney_bug.cfm :: 9	10.015
C:\Work\...\cfthread_a.cfm :: 4	10.015
C:\Work\...\cfthread\junk.cfm :: 2	2.000
C:\Work\...\cfthread\cfthread.cfm :: 1	1.524
C:\Work\...\cfthread\exchThread.cfm :: 1	1.141
C:\Work\...\cfthread\hal.cfm :: 24	1.000

Left sidebar menu: REQUEST STATISTICS (Active Requests, Active ColdFusion Threads, Slowest Requests, Slowest ColdFusion Threads, Active Sessions, Cumulative Server Usage, Highest Hit Counts, Template Cache Status, Request Throttle Data), MEMORY USAGE, DATABASE, ERRORS.

Summary

```
<cfloop query="dir">
  <cfset threadname = "thread_#i++#">
  <cfthread name="#threadname#" filename="#dir.name"
    <cfset source = "#src#\#filename#">
    <cffile action="COPY" source="#source#"
      destination="#dest"
      <!--- Set variable in THREAD scope --->
      <cfset thread.newpath =
        "#dest#\#attributes.filename#">
      Copied Successfully
    </cfthread>
  </cfloop>
  <!--- Access THREAD scope outside --->
  <cfset reading=thread_1.newpath>
  <cfoutput>#thread_1.output#</cfoutput>
```

Pass as
attribute to
thread

Local to the
thread

Set the result
in Thread
scope

Access using
Attributes
scope

Access data
from Thread
scope

Access Output
from thread

Questions ?

rukumar@adobe.com

<http://coldfused.blogspot.com>

Better by Adobe.™