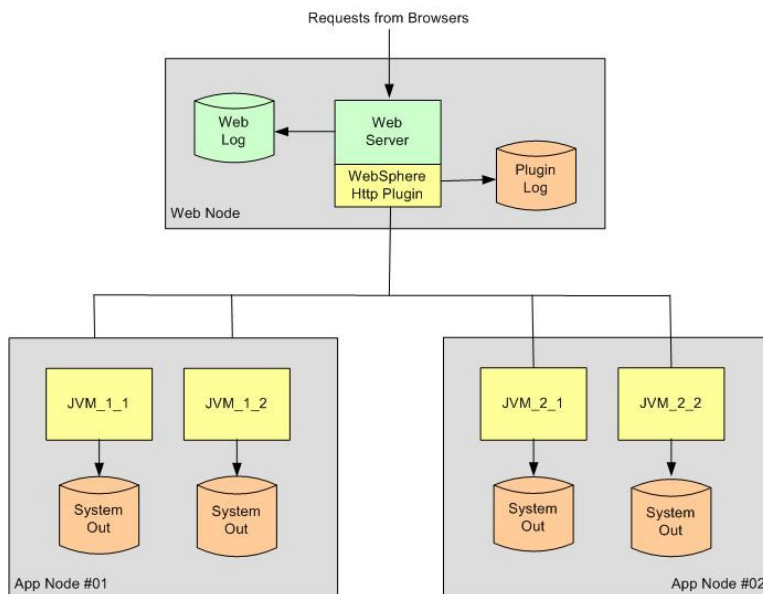

MONITORING APPLICATION PERFORMANCE USING WEBSPHERE REQUEST METRICS

Ken Gottry
August, 2007

1 OVERVIEW

WebSphere v6 introduced Request Metrics (PMRM), which unlike Performance Monitoring Infrastructure (PMI) metrics, are transaction based. PMRM can be a useful first step in performance analysis of your application. The PMRM records show the elapsed time for each request. It is also possible to write multiple PMRM records for a single request, each record showing the elapsed time within a single hop within the transaction (e.g. a JDBC hop or an EJB hop).

The PMRM records are written to the SystemOut log file for the JVM in which the request is processed. In a Network Deployment configuration, the WebSphere Http Plugin running inside the web server also writes PMRM records to its `http-plugin.log` file giving a composite view of application performance across all JVM's.



Each record in the log file contains considerable information about the flow of the transaction through the WebSphere JVMs. The last fields are of particular interest during performance monitoring. The SystemOut file contains

- Name of servlet

- Response time

In addition, the web server http-plugin.log file contains

- Size of the request
- Size of the response

PMRM record in SystemOut.log

```
[8/10/07 15:10:10:630 EDT] 000000ad PmiRmArmWrapp I PMRM0003I:  
parent:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1 -  
current:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1  
type=URI detail=/acme/aOrgMovePopup.do elapsed=58220
```

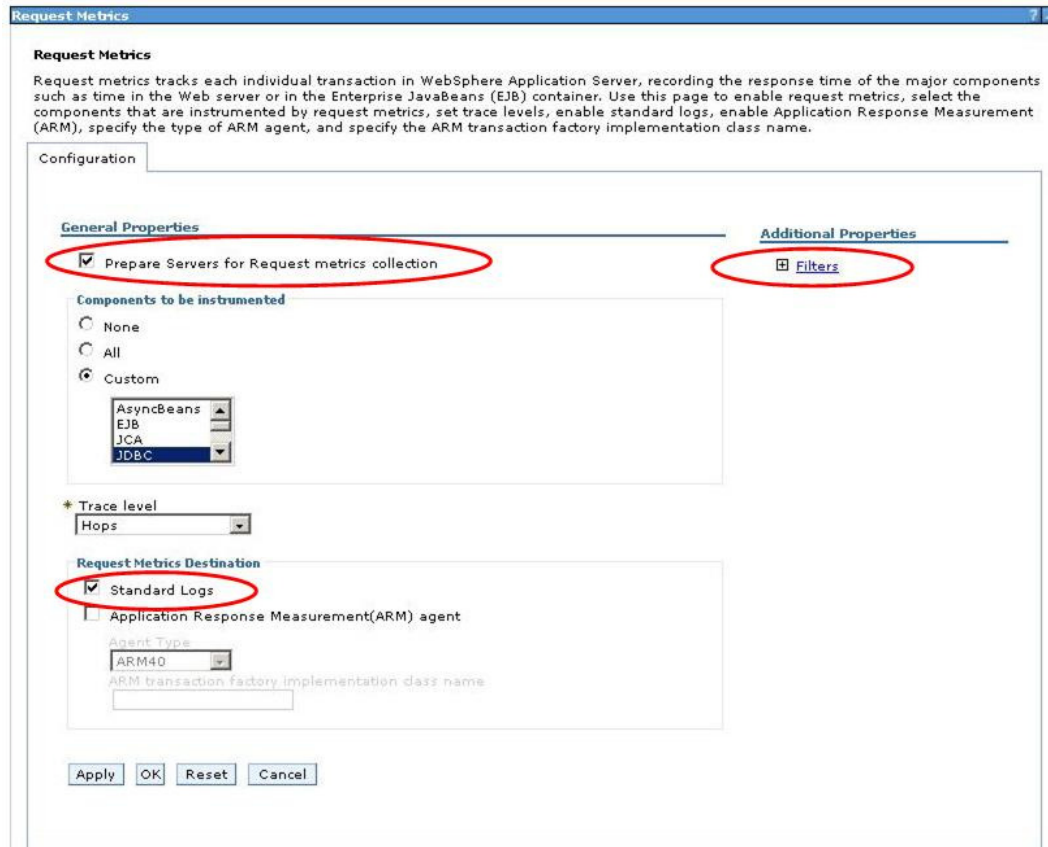
PMRM record in http-plugin.log

```
[Wed Dec 20 17:54:57 2006] 000065f3 000001b2 - PLUGIN: parent:ver=1,  
ip=172.22.23.55, time=1163796088727, pid=26099, reqid=360424, event=1 -  
current:ver=1, ip=172.22.23.55, time=1163796088727, pid=26099, reqid=360424, event=1  
type=HTTP detail=/acme/AdminMaint.do elapsed=318 bytesIn=0 bytesOut=4551
```

The remainder of this article and the enclosed scripts apply to the Request Metrics records found in the SystemOut log file of an individual JVM unless otherwise noted. All scripts can be easily adapted to the slightly different format of the PMRM records in the http-plugin log file.

2 ACTIVATING REQUEST METRICS

Navigate to Request Metrics under Monitoring and Tuning on the Admin Console. Check the box to prepare the servers for PMRM collection, identify the type of PMRM records to collect, and indicate the PMRM records should be written to SystemOut. It is possible to establish filters to control for which requests you want to collect PMRM records.



3 A SMALL SCRIPT TO ANALYZE REQUEST METRICS

The scripts in this section allow you to use PMRM records to analyze the performance of your application. You can determine

- Transaction profile mix
- Transactions per second
- Response time, average and individual

In a network deployment environment, you can use the PMRM records in the http-plugin log file to determine also

- Response size, average and individual

True, similar information can be gleaned from standard web access logs. However, as you will see later in this article, this is only the beginning of the performance information contained in the PMRM records.

3.1 Using PMRM records in SystemOut

Below is a subset of PMRM records from a WebSphere log. To focus attention, only a handful of pertinent data fields from each PMRM record are shown on each line.

```
type=URI detail=/acme/AdminMaint.do elapsed=318
type=URI detail=/acme/TransSearch.do elapsed=3152
type=URI detail=/acme/UserAuth.do elapsed=2082
type=URI detail=/acme/UserAuth.do elapsed=69
type=URI detail=/acme/Login.do elapsed=23997
type=URI detail=/acme/Login.do elapsed=359
type=URI detail=/acme/Login.do elapsed=502
```

Below is a simple Awk script to analyze the PMRM records in a SystemOut file. Note that record selection is done via a couple of preceding grep commands so the Awk script simply processes all input records

```
grep "PMRM0003I" SystemOut.log | \
  grep "type=URI" | \
  awk -f PMRM.awk | \
  sort -k4 -k1 -k2
```

Figure 1 - PMRM.sh shell script

```
{
DATE=substr($1,2)
  split($2,T,":")
  if (length(T[1]) == 1)
    T[1]=sprintf("0%s", T[1])
  TIME=sprintf("%s:%s:%s:%s", T[1], T[2], T[3], T[4])
  MS=substr($13,9)
  URL=substr($12,8)
  printf("%-8s\t%-10s\t%-10d\t%-s\n", DATE, TIME, MS, URL)
}
```

Figure 2 - PMRM.awk script to process PMRM records in SystemOut.log files

Below is a sample of the output from this Awk script

```
8/10/07      15:10:10:630  58220   /acme/aOrgMovePopup.do
8/10/07      11:44:42:416  53172   /acme/runquery.do
8/10/07      15:08:15:276  32480   /acme/userAuth.do
8/10/07      13:22:38:056  25176   /acme/TRRSubmit.do
8/10/07      12:03:23:192  24279   /acme/avPayment.do
8/10/07      04:25:26:274  22452   /acme/runquery.do
8/10/07      02:33:38:575  21257   /acme/runquery.do
8/10/07      16:05:30:415  20004   /acme/aOrgConRef.do
8/10/07      12:11:01:407  18115   /acme/aOrgConRef.do
```

3.2 Using PMRM records in http-plugin.log

The PMRM records in the `http-plugin.log` file also contain the number of bytes of the request and of the response.

```
type=HTTP detail=/acme/AdminMaint.do elapsed=318 bytesIn=0 bytesOut=4551
type=HTTP detail=/acme/TransSearch.do elapsed=3152 bytesIn=0 bytesOut=116546
type=HTTP detail=/acme/UserAuth.do elapsed=2082 bytesIn=73 bytesOut=4577
type=HTTP detail=/acme/UserAuth.do elapsed=69 bytesIn=0 bytesOut=5358
type=HTTP detail=/acme/Login.do elapsed=23997 bytesIn=0 bytesOut=3323
type=HTTP detail=/acme/Login.do elapsed=359 bytesIn=0 bytesOut=3323
type=HTTP detail=/acme/Login.do elapsed=502 bytesIn=0 bytesOut=3323
```

This Awk script separates static content (e.g. GIF, JPG) from dynamic content (e.g. servlets, ASP and JSP pages). The former is considered “embedded content” and is simply totaled at the end. Unlike the previous Awk script, this one uses Awk commands to select records

Note: In a production environment static content is usually rendered by a front-end web server and would not appear in the WebSphere SystemOut log.

```
BEGIN {
    printf("%-10s%10s%10s\t%s\n", "Time", "Resp(ms)", "Bytes", "URL")
    printf("%-10s%10s%10s\t%s\n", "----", "-----", "-----", "----")
}
{
    if (substr($10,1,10) != "parent:ver")
next
TIME=$4

    # type=HTTP detail=/acme elapsed=1455 bytesIn=0 bytesOut=0
    TYPE=substr($13,6)
    if (TYPE == "HTTP" || TYPE == "URI") {
        NBR=split($14,U,".")
        if (NBR > 0) {
            EXT=tolower(U[NBR])
            if ( (EXT=="do") || (EXT=="asp") || (EXT=="jsp") )
                NBR=0
        }
    }

    if (NBR == 0) {
        DETAIL=substr($14,8)
        MS=substr($15,9)
        OUT=substr($17,10)
        printf("%-10s%10d%10d\t%s\n", TIME, MS, OUT, DETAIL)
        TRX++
    } else {
        OBJ++
    }
} else {
    } else {
printf("TYPE=%s\n", TYPE)
OTH++
    }
}
END {
```

```

printf("%-10s%10s%10s\t%s\n", "----", "-----", "-----", "----")
printf("%-10s%10s%10s\t%s\n", "Time", "Resp(ms)", "Bytes", "URL")
printf("%-10s%10s%10s\t%s\n", "----", "-----", "-----", "----")

printf(" #####\n")
printf(" Trans\t%10d\n", TRX)
printf(" Embed\t%10d\n", OBJ)
printf(" Other\t%10d\n", OTH)
}

```

Below is a sample output from this Awk script. The summary at the end says there were 7 requests for servlets and 13 requests for embedded content such as GIF and JPG files.

```

Time          Resp(ms)      Bytes URL
-----
10:49:16 359          3323 /acme/Login.do
10:55:24 502          3323 /acme/Login.do
12:23:08 2082         4577 /acme/UserAuth.do
12:49:26 69           5358 /acme/UserAuth.do
15:15:33 23997        3323 /acme/Login.do
17:54:57 318          4551 /acme/AdminMaint.do
14:34:56 3152        116546 /acme/TransSearch.do
-----
Time          Resp(ms)      Bytes URL
-----
#####
Trans          7
Embed         13
Other          0

```

4 USING REQUEST METRICS TO MONITOR JDBC ACTIVITY

4.1 Overview

There are many tools to drill down into the performance of individual requests within a WebSphere application server. CA/Wily Introscope is one of my favorites, while Glassbox is a nice Open Source alternative. Request Metrics can provide similar, albeit less detailed, analysis using a few simple scripts.

4.2 Locating all PMRM records for a specific request

To begin, via the Request Metrics page on the Admin Console, ensure you have specified All or Custom as the type of components to be instrumented.

Step 1: Identify long-running requests – Produce the list of response times of individual requests using the Awk script described above. In the sample output below, the aOrgMovePopup servlet ran for over 58 seconds.

```

8/10/07      15:10:10:630  58220  /acme/aOrgMovePopup.do
8/10/07      11:44:42:416  53172  /acme/runquery.do

```

8/10/07	15:08:15:276	32480	/acme/userAuth.do
8/10/07	13:22:38:056	25176	/acme/RevSubmit.do
8/10/07	12:03:23:192	24279	/acme/avPayment.do
8/10/07	04:25:26:274	22452	/acme/runquery.do
8/10/07	02:33:38:575	21257	/acme/runquery.do
8/10/07	16:05:30:415	20004	/acme/aOrgContentRefresh.do
8/10/07	12:11:01:407	18115	/acme/aOrgContentRefresh.do

Step 2: Breakdown long-running request into its components – The Korn shell and Awk scripts at the end of this section locate the PMRM records associated with the long running aOrgMovePopup. Here’s how.

First, the script looks for the PMRM record for this request. It is type=URI and ends with elapsed=58220. The “reqid” of the “current” portion of this PMRM record uniquely identifies the PMRM records associated with the aOrgMovePopup request.

```
[8/10/07 15:10:10:630 EDT] 000000ad PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1 -
current:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1 type=URI
detail=/acme/aOrgMovePopup.do elapsed=58220
```

Next, all PMRM records with reqid=22912 are extracted from the SystemOut file and formatted. In the following output there was one prepared statement that ran for 47.982 seconds and another than ran for 9.746 seconds. These SQL calls consumed 82% and 17% respectively of the total response time of the aOrgMovePopup request.

The SQL that ran for over 47 seconds must have started shortly after 15:09:12:863, the ending time of the preceding PMRM record

15:09:12:416	0	javax.resource.spi.ManagedConnectionFactory. matchManagedConnections(Set,Subject,ConnectionRequestInfo)
15:09:12:424	1	java.sql.Connection.rollback()
15:09:12:427	0	javax.resource.spi.ManagedConnectionFactory. getConnection(Subject, ConnectionRequestInfo)
15:09:12:430	0	javax.resource.spi.LocalTransaction.begin()
15:09:12:440	7	java.sql.PreparedStatement.executeQuery()
15:09:12:863	418	java.sql.PreparedStatement.executeQuery()
15:10:00:851	47982	java.sql.PreparedStatement.executeQuery()
15:10:10:606	9746	java.sql.PreparedStatement.executeQuery()
15:10:10:616	4	javax.resource.spi.LocalTransaction.commit()
15:10:10:620	2	javax.resource.spi.ManagedConnectionFactory.cleanup()
15:10:10:630	58220	/sam/aOrgMovePopup.do

Step 3: Scan SystemOut for long-running SQL –Scanning the SystemOut file for the end time of the preceding SQL, you can locate the Hibernate record at that time to obtain the SQL that ran for 47.982 seconds.

If multiple users are executing requests simultaneously, use the web container thread ID (000000ad in this example) to match Hibernate and PMRM records.

```
[8/10/07 15:09:12:863 EDT] 000000ad PmiRmArmWrapp I PMRM0003I:
```

```
parent:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1 -
current:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22918,event=1
type=JDBC detail=java.sql.PreparedStatement.executeQuery() elapsed=418
```

```
[8/10/07 15:09:12:869 EDT] 000000ad SystemOut      O Hibernate:  select
client0_.TRSACT_PK as col_0_0_ from TRS trsId0_ where upper(client0_.NAME)=?
```

```
[8/10/07 15:10:00:851 EDT] 000000ad PmiRmArmWrapp I PMRM0003I:
parent:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22912,event=1 -
current:ver=1,ip=3.142.17.18,time=1186760144534,pid=6136,reqid=22919,event=1
type=JDBC detail=java.sql.PreparedStatement.executeQuery() elapsed=47982
```

4.3 Scripts

The scripts in this section show one way to extract all PMRM records for a specific request from a SystemOut file. The Korn shell script accepts the elapsed milliseconds of the request as its single argument. Grep locates the single PMRM with the specified elapsed time and the req.awk script extracts the request ID. The elapsed.awk script then formats and prints all PMRM records with a matching request ID.

```
# arg1 is the elapsed time of the request you want to analyze
MS=$1
# define location of files and scripts
HOME_DIR=/export/home/kgottry/bin
AWK1=${HOME_DIR}/elapsed.awk
AWK2=${HOME_DIR}/reqid.awk
TMP=/tmp/t.$$
JVMDIR=/appbin/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/JVM_1_1
# get name of SystemOut log containing desired PMRM record
grep "elapsed=${MS}" ${JVMDIR}/SystemOut*log | head -1 > ${TMP}
FILE=`awk -f ${AWK1} ${TMP} | awk '{print $1}'`
# get the request ID of the PMRM record for the URI
REQID=`awk -f ${AWK1} ${TMP} | awk '{print $2}'`
# print all the PMRM records with matching request ID
grep ${REQID} ${FILE} | awk -f ${AWK2}
rm ${TMP} 2>/dev/null
```

```
# req.awk - extracts the request ID from the URI record
{
    TYPE=substr($1,6)
    POS=index($1, ":")
    DATE=substr($1,POS+2)
    split($2,T,":")
    if (length(T[1]) == 1)
T[1]=sprintf("0%s", T[1])
    TIME=sprintf("%s:%s:%s:%s", T[1], T[2], T[3], T[4])

    POS1=index($0,"detail=")
    POS2=index($0, "elapsed=")
    DETAIL=substr($0, POS1+7, POS2-POS1-8)
    POS3=index( substr($0, POS2+8), " ")
    if (POS3 > 0)
        MS=substr($0,POS2+8, POS3-POS2+9)
    else
        MS=substr($0,POS2+8, length($0)-POS2+8)
```

```
printf("%-8s\t%-10s\t%10d\t%s\n", DATE, TIME, MS, DETAIL)
}
```

```
# elapsed.awk - format and print all PMRM records with matching request ID
{
    # multi-file grep prepends filename to each matching line
    NBR=split($1,F,":")
    if (NBR < 2)
        FILE="unknown"
    else
        FILE=F[1]
    POS1=index($0,"reqid=")
    POS2=index(substr($0,POS1),",")
    printf("%s %s", FILE, substr($0, POS1, POS2-1))
}
```

5 CONCLUSION

WebSphere v6 introduced request metrics. Using simple scripts these PMRM log records can provide transaction-level performance analysis of your application. The PMRM log records can also show performance breakdown by component, such as JDBC or EJB, for each request.