Fathom – Beyond Out-of-the-Box

In Progressions 56, I presented an article on implementing Fathom Management in a fairly large AIX environment. A year later, I'm happy to report that Fathom has been doing a fairly decent job of making me look good. System uptime is exceptional. Performance is great. In a nutshell, the users are happy, and when the users are happy, the V.P. is happy…and when the V.P. is happy, I get my contract renewed!

Naturally, the next question we ask is "What else can we do with Fathom?" It's monitoring my Progress databases and AppServers, emailing and/or paging me if there's a problem, and just generally keeping an eye on everything. The answer is simple: let's see if it can help us improve our existing systems. Before we do that, we need some additional tools to simplify the creation of Fathom jobs.

The Multi-DB Template

Those of you that read the article in Progressions 56 may remember the multi-db template from one of my examples. Today I'd like to go into it in more detail.

One of the fun things about Fathom is that it allows you to define jobs that will act on multiple databases. In order to help you do that, it sets certain environment variables with comma-separated lists of values. For example, if you create a job that will act on databases /usr/db/sports and /usr/db/demo, Progress will set the variables RESRC_DBNAME and RESRC_DBPATH as follows:

RESRC_DBNAME= "sports,demo"
RESRC_DBPATH="/usr/dbsports,/usr/db/demo"

That's all fine, but not very useful unless you break out the values into more useable arrays. This is where the multi-db template comes in:

```
#!/bin/ksh
SHELL=/bin/ksh;export SHELL
# FUNCTIONS
########################################
#
# ValidateDBList: Validates passed env
#                 vars that contain DBs
#                 to process
#
#
########################################
ValidateDBList()
{
  DBLIST=$RESRC_DBNAME
  DBPATHLIST=$RESRC_DBPATH
  OLDIFS=$IFS
  IFS=","

  cnt1=0; cnt2=0
```

```
  for i in $DBLIST
  do
    DBNAME[cnt1]=$i
    ((cnt1=cnt1+1))
  done
  for i in $DBPATHLIST
  do
    DBPATH[cnt2]=$i
    ((cnt2=cnt2+1))
  done

  IFS=$OLDIFS
}
########################################
#
# MAIN
#
########################################

# Fathom passes the db names and paths
# in two vars. ValidateDBList puts the
# db names and paths in two arrays
# DBNAME and DBPATH
ValidateDBList

i=0
while (( i < ${#DBNAME[*]} ))
do
   # Extract the current values
   # Some of the Fathom utilities are
   # hard-coded to act on RESRC_DBPATH
   # and RESRC_DBNAME.
   # This way we standardize how we
   # extract single DBs from the list
   RESRC_DBNAME=${DBNAME[$i]}
   RESRC_DBPATH=${DBPATH[$i]}

   # INSERT YOUR CODE HERE

   ((i=i+1))
done
```

I've cut the template down to as bare as possible in order to save space but it's basically ready to use as-is. A more complete template with some added bells and whistles is available – just email me.

Now that we have the tools, let's put them to use.

User-Controlled Locks and Transactions

In my perfect world, all applications are loosely-coupled, service-oriented n-tier systems with the application logic gracefully separated from the user interface. The Progress Reference Architecture is the revered bible and all non-conforming programs are busily being converted at this very moment.

Then I wake up.

I wake up and realize that there are still thousands of users out there who actually control the scope of a lock and transaction. I swear, they can actually get up and go for a cup of coffee while (gasp!!!) an open and active transaction sits there at their desk. I know some of you don't believe me but really, it's true!

So what does an organization do? Do they budget a gazillion dollars to rewrite all their user interface code? Do they throw out their old application and buy a new one? Ok. Sure. These are both viable solutions. But what do you do right now? You monitor, of course.

Monitoring Lock Conflicts

Take the multi-db template and insert the line:

```
_progres -b $RESRC_DBPATH -p dbdeadlocks.p
```

where it says "INSERT YOUR CODE HERE." Schedule the new script as a Fathom job and voila! You are monitoring database lock conflicts. The heart of the dbdeadlocks.p is quite simple and is provided below. As with the multi-db template, email me for more complete code examples.

```
FOR EACH DICTDB._lock NO-LOCK:
  IF _lock-recid = ? THEN LEAVE.
  FIND DICTDB._file WHERE _file-number = _lock-table NO-LOCK.
  FIND DICTDB._Connect WHERE _Connect-usr = _lock-usr NO-LOCK.

  /* In this particular application there is a way to see what
     actual function the user is running in the application
  */
  IF _connect-type = "SELF" THEN
      RUN GetSidmaFcn(_lock-name, _connect-device, OUTPUT FcnName).

  DO TRANSACTION:
    CREATE ttLocks.
    ASSIGN UsrID     = _lock-usr
           UsrName   = _lock-name
           UsrPID    = _connect-pid
           UsrTx     = _connect-transid
           UsrSvr    = _connect-Device
           TableName = _file-name
           LKRecid   = _lock-recid
           LKflags   = REPLACE(_lock-flags," ","")
           LKConflict = (IF _lock-flags MATCHES "*Q*" THEN yes
                                                      ELSE no)
           SidmaFcn  = FcnName.
  END.
END.
```

Once the temp-table is built, all that's left to do is to go through ttLocks and output the information that will help the support group resolve the lock conflict:

```
/** Look at each lock entry that is in conflict **/
for each ttlocks NO-LOCK where LKConflict BREAK BY LKConflict :

  IF NOT FIRST-OF(LKConflict) THEN NEXT.

  /** Look at ALL client sessions that are using the record in conflict
        That is, both the conflict locks and the non-conflict locks
  **/
  for each bfttlocks where bfttlocks.lkrecid = ttlocks.lkrecid:

     put bfttlocks.UsrName
         bfttlocks.UsrPID "  "
         bfttlocks.TableName " "
         bfttlocks.LKRecid "  "
         bfttLocks.LKFlags "    "
         bfttlocks.SidmaFcn
         skip.

    ConflictPIDS = ConflictPIDS + string(bfttlocks.UsrPID) + ",".
  end.
end.
put skip(1).

if num-entries(conflictpids) = 0 then quit.

put unformatted "************** CONFLICT FOUND **************" skip.

do Cnt = 1 to num-entries(conflictPIDS) - 1:
  os-command silent ps -ef | grep value(entry(Cnt,ConflictPIDS)) | grep
  -v grep.
end.
put skip(2).
quit.
```

Finally, all the script has to do is validate if the Progress program outputted any data and either email support, raise a Fathom alert or both.

Monitoring Long Transactions

Monitoring long transactions is a little more complicated, especially if multi-database transactions are involved. Experience has taught me that I have to make sure I limit the number of alerts I generate. If I generate too many, and especially if many or most are false alarms, I will inadvertently train the support staff to ignore Fathom-generated alerts. Of course the opposite is also true: if Fathom does not generate alerts when it's supposed to then why bother implementing Fathom at all?

The solution to this problem I will save for the next issue of Progressions.

Conclusion

Take it from a guy who's spent some time with Oracle Enterprise Manager 9i and 10g and IBM's Tivoli Monitoring for databases: Fathom is a very cool product. The value out of the box is irrefutable: monitoring, trending, alerts… All these features are quite

good and every release is only getting better. Are there still bugs? Yes, unfortunately. The product is still growing and maturing. But so far none of the bugs I've faced have been show-stoppers.

So whether you're in the midst of deciding whether or not to implement Fathom in your environment, or if you're already in the middle of your Fathom implementation, I hope you find the extra functionality presented here useful. And as always, feel free to email me with any questions you may have.

Paul Koufalis
pk@progresswiz.com

Blurb:

Paul began his Progress career in 1994 after finishing a computer engineering degree at McGill University. He started Progresswiz in 1999 and has been working as a Progress DBA, UNIX admin and all-round technical consultant ever since. If you happen to be in Lisbon for PTW 2005 be sure to sit in on one or both of his sessions: "Fathom in the Real World" and "How do I Kill Thee: Let me Count the Ways…"