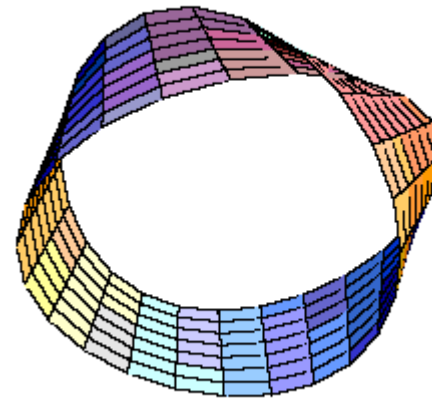


APSY and Saved Compilations



Alex Kodat
Sirius Software Inc.

User Language

- Formerly procedural, now an object-oriented language
- English-like syntax
- Compiled into binary format before being executed
 - Not into machine language like some compilers
 - Binary format is called *quads*
 - Because originally every quad was four 32-bit words
 - Now quads vary in number of 32-bit words
 - A User Language statement can compile to one or more sets of quads
 - Compiled code includes compiled DML
 - So no separate DML compilation ala SQL



Advantages of Compiling into Quads

- Quads more compact than machine code
 - A compiled quad can do the equivalent of hundreds of machine language instructions
- Quads can do run-time optimization which is much more difficult when compiled to machine code
- Debugging and development easier than compiling to machine code
- Execution much faster than pure interpretation
 - Parsing and tokenizing only needs to be done once
- This is the general direction of the industry
 - Compiling to p-code (pseudo-code)
 - Java compiles to byte-code
 - VB.Net
 - So UL designers (luckily?) picked the correct design



UL Compiler

- Saves quads in QTBL
 - And variables in STBL and VTBL
 - And a few other things in other tables
- Quads are ready to run
- Quads are not relocatable
- Quads and variables cannot be saved to disk
 - Because generated quads are file/field specific
 - If a field changes from ORDERED to KEY, for example, the quad might change
 - If number of segments in a file changes, variables might change
 - This is all probably a historical mistake
 - *Just in time* compilation



Advantages of Just In Time Compilation

- One less thing to manage
 - No object code to manage
 - No danger of losing the source
 - Less confusion about what you're running
- Compiler and evaluator can be easily changed between releases or even in a ZAP
 - No re-compilation step required
 - No need to support old format quads
- Development simplified
 - No extra compile-link steps



Disadvantages of Just In Time Compilation

- Compilation can be expensive
 - Parsing and tokenizing
 - Looking up variable names
 - Generating code
 - Optimizing generated code
 - Not atypical for compile time to be greater than run time for most procedures
- Users need to have files/groups required for compilation open in mode required by application
 - Which means users can access files/groups themselves
 - A security hole



Non-relocatability of UL Quads

- Means entire application must be loaded at once
 - Impractical, given server sizes and size of typical applications
- Procedure need a mechanism to navigate among themselves
 - Convenient for conversational 3270 applications
 - And not bad for others
 - But a hassle if one wants a call mechanism



Application subSYstems – AKA APSY

- Provides a mechanism for saving and reusing compilations in a particular Online run
- Provides a mechanism for transferring between procedures
- Provides a mechanism for application access to files and groups while protecting them from end-users
- Why not APSUB or APSYS?
 - Maybe because the internal module for managing application subsystems was called APSY
- And more



An Application Subsystem Definition

- Operation described in special Model 204 file called CCASYS
 - Files/groups used by subsystem defined there
 - Names of special procedures defined there
 - Other subsystem processing described there
- User Language for subsystem is in a special procedure file or group indicated in CCASYS



Starting a Subsystem

- Done with START SUBSYSTEM <subsystem> command
 - If user issues a <subsystem> command 204 looks in CCASYS for the subsystem name
 - If autostart, 204 does an implicit START SUBSYSTEM <subsystem>
 - So every typo does a CCASYS lookup
 - New Model 204 commands will preempt subsystems
- In memory tables built for subsystem
 - Subsystem operation settings
 - File/group use tables for all files/groups defined to subsystem
 - Procedure dictionary of all procedures with pre-compiled and non-pre-compiled prefixes
 - Procedures in procedure file or locked procedure group files are “locked” - not allowed to be updates



Running APSY procedures

- The *Next* global variable indicates the next outer procedure to be run
 - The procedure must have a non-pre-compiled or pre-compiled procedure prefix
 - Prefixes are part of subsystem definition
- If prefix is a non-pre-compiled prefix, the procedure is INCLUDED as if someone typed an INCLUDE command
 - User Language compilation **never** saved, no matter what
- If prefix is a pre-compiled prefix, special processing occurs
- If procedure not found (or name invalid), it's off to error procedure



Running a Pre-Compiled Procedure...

The “good old days”

- Check for saved compilation
 - If there use it... more on this later
 - Groups can make this more complicated... more on this later
 - If not...
- Flag is set indicating pre-compiled procedure
- Procedure is INCLUDED
- In simple cases, after the compilation is done, the compilation is saved
 - But sometimes it can't be



Can't Save a Pre-Compiled Procedure...

The “good old days” (cont...)

- If command found before Begin, the compilation could not be saved
- If command found after Begin, the compilation was saved... and then discarded
 - This would include the case of a second Begin



Can't Save a Pre-Compiled Procedure... The “good old days” (...cont...)

- If group context for the procedures and an INCLUDE from an unlocked file is detected, the compilation cannot be saved
 - NUMLK is a procedure group parameter in the subsystem definition that indicates how many members of the group are considered “locked”
 - The NUMLK **last** files in the group are “locked”
 - When subsystem started, APSY places a lock on all procedures with the pre-compiled or non-pre-compiled prefix in all locked files in the group
 - So no one can update those procedures
 - But, INCLUDED procedures are not locked



Can't Save a Pre-Compiled Procedure... The “good old days” (...cont...)

- Oddly, INCLUDE from file not in the subsystem's procedure group does **not** prevent saving the compilation
 - Was this intentional?
 - File does have to be opened by subsystem
- If reference to database file/group that is not in the subsystem definition, compilation cannot be saved
 - Database file reference is any In clause
 - `in file foo find` for example
 - `%foo is object recordset in file foo` for example
 - Check done after compilation so can be difficult to find the bad reference



Running a Saved Compilation

- Make sure saved compilation usable
 - Mainly a group consistency issue
 - More on this later
- Load the tables from CCATEMP (or elsewhere)
 - QTBL – Often big. Contains the code (quads)
 - NTBL – Moderately big. Contains hashes of variable names
 - Kind of a waste as it's rarely used at run-time
 - Maybe some day this could be reduced/eliminated
 - VTBL – Can be big. Contains variables and structures used for evaluation
 - STBL – Can be big. Contains string values and arrays (initial values loaded)
 - FTBL – Usually not big. Contains group layouts and cache of group fields used in request
 - Must be merged with what's already there so a bit complicated



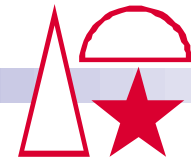
Performance of Saved Compilation Load (APSY Load)

- Several orders of magnitude faster than a compilation
- But, for large requests, can still be significant
 - Might require hundreds of logical disk reads (DKPRs) for CCATEMP pages
 - Often from buffer pool, but sometimes from disk
 - Data moved with moderately expensive character stream move instruction (MVCL)
 - Move this (large) stream of data from here to there



Improving APSY Load Performance - APSYPAGE

- APSYPAGE sets number of 4K (as opposed to 204's traditional 6184-byte) pages to be used to hold saved compilations
- 204 saves as many requests as it can in the APSYPAGE pool
 - In 6.1 the pages are also kept in CCATEMP as a backup
 - So MRU requests could be saved in APSYPAGE pool
 - But, unfortunately, doubles page requirements
- Pages from APSYPAGE pool use direct memory access instead of CCATEMP logical reads
- Data moved from APSYPAGE pool to server using very fast MVPG (MoVe PaGe) instruction



Using MVPG During APSY Load

- Requires both source and target be hardware page-aligned (4K boundaries)
 - APSYPAGE pool maintained as hardware pages
 - If APSYPAGE used, QTBL, VTBL, NTBL and STBL are page-aligned in servers
 - Because these are the big tables that are APSY loaded
 - Can require an extra 4088 bytes of server per table in worst case
 - So, if enabling APSYPAGE, increase SERVSIZES by at least 16352
- Movement is highly hardware/microcode optimized
 - Sometimes happens asynchronously
 - Sometimes handled by special-purpose processors rather than the CPU
 - So, a big win, all around



Reducing the Amount of Data Loaded

- QTBL and NTBL almost always read/only at run-time
- So, why not use a shared copy instead of APSY loading them?
- Set RESSIZE to make this possible
- Set RESLTHR to have frequently loaded requests use resident/shared QTBL and NTBL
 - “Traditional” parameter of RESTHRSH reduce amount of data server swapped (shared part not swapped)
 - But in many modern configurations APSY loads are much more common than server swaps
- Most sites can easily afford to burn a little memory to save CPU
 - RESSIZE=40000000 and RESLTHR=100 probably a decent starter most places



Saved Command in Pre-Compiled Procedures

- Intended to solve the problem of wishing to put commands at the start of a pre-compiled procedure
 - Most commonly, a UTABLE command
 - Occasionally a RESET or USE
- Also, intended to solve the problem of wishing to put commands after a pre-compiled procedure
 - Again, UTABLE or RESET commands might be common
 - Command is often partially dynamic, with dynamic part in a dummy string global set by the pre-compiled request
 - `UTABLE LQTB L ?&OLDQTB L`
- Old technology was to transfer to intermediate non-pre-compiled procedure



How Do Saved Commands in Pre-Compiled Procedures Work?

- After pre-compiled procedure is INCLUDED, APSY examines every line in the procedure until it hits the Begin
- Any command before the Begin is saved into CCATEMP
 - Basically a special temporary procedure (like proc 0)
 - Call this the pre-command procedure
- After the compilation is done, APSY goes back to examining every line in the procedure until it hits the End
 - Again, a special temporary procedure
 - Call this the post-command procedure



Running Saved Requests in Model 204 V6.1 and Later

- Pre-command procedure is INCLUDED, if present
- Saved request is loaded and evaluated
- Post-command procedure is INCLUDED, if present
- Seems simple enough



Problem One with Saved Commands in Pre-Compiled Procedures

- What if there's another Begin after the saved request?
- Second UL program simply saved as “commands”?
- No, request not saved



Problem Two with Saved Commands in Pre-Compiled Procedures

- Should commands be saved before or after dummy string (?&<mumble>) substitution?
- Well, since usually the desire is for the dummy strings to be able to vary at run-time...
- Commands saved before dummy string substitution
- So substitution done when pre-command and post-command procedures are processed by each user



Problem Three with Saved Commands in Pre-Compiled Procedures

- What about comments?
- We don't want to waste CCATEMP and CPU time to process comments
- Especially since a pre-compiled proc might have a large block of commands before the Begin statement
- So comments are not saved in a pre-compiled request
 - Code has to check for *SLEEP, *LOWER, *UPPER, etc.



Problem Four with Saved Commands in Pre-Compiled Procedures

- The interaction of Begin/End detection and dummy strings
- Detection **can** be fooled
- But don't bother, it's going to be too flakey



Extra Begin/End Blocks in Pre-Compiled Procedures: The **Wrong** Way

```
?&DEBUG b  
?&DEBUG printtext Here I am  
?&DEBUG end  
b  
...
```



Extra Begin/End Blocks in Pre-Compiled Procedures: The **Right** Way

```
?&DEBUG INCLUDE EXTRA  
b  
    . . .  
end
```

Contents of procedure Extra:

```
b  
printText Here I am  
end
```



Understanding Pre-Compilation

- Spend time looking at the journal
 - SirScan is an ideal tool for this
- Look especially at the since-last records
 - LAST='CMPL' for compilation
 - LAST='LOAD' for APSY loads
 - If suppressed can be inferred from EVAL with no CMPL
 - LAST='EVAL' for evaluation
- Funny stats for
PROC-FILE='CCASYS' PROC='
are for internal CCASYS procs usually run at entry to subsystem



Looking at the Journal

LI MAINT DLIST

```
ST USERID='ALEX' ACCOUNT='ALEX' LAST='EVAL' SUBSYSTEM='MAINT'  
  PROC-FILE='CCASYS' PROC='          ' GTBL=118 STBL=63 TTBL=5 FINDS=1 RECDs=1  
  RQTM=1 DKPR=10  
MS M204.1168: IN FILE MAINT INCLUDE MAINTN_START  
ST USERID='ALEX' ACCOUNT='ALEX' LAST='CMPL' SUBSYSTEM='MAINT' PROC-FILE='MAINT'  
  PROC='MAINTN_START' NTBL=11 QTBL=76 STBL=594 VTBL=32 PDL=540 CPU=2 PCPU=1000  
  RQTM=2  
ST USERID='ALEX' ACCOUNT='ALEX' LAST='EVAL' SUBSYSTEM='MAINT' PROC-FILE='MAINT'  
  PROC='MAINTN_START' NTBL=11 GTBL=186 QTBL=76 STBL=607 VTBL=32 PDL=540 RQTM=1  
MS M204.1168: IN FILE MAINT INCLUDE MAINTP_DLIST  
ST USERID='ALEX' ACCOUNT='ALEX' LAST='EVAL' SUBSYSTEM='MAINT' PROC-FILE='MAINT'  
  PROC='MAINTP_DLIST' GTBL=243 STBL=21610 TTBL=5 PDL=560 CPU=4 OUT=300 FINDS=3  
  SORTS=1 RECDs=2 PCPU=800 RQTM=5 BXFIND=8 DKPR=264  
AD MSIR.0359: WEB status 200 OK  
AD MSIR.0762: Sending 10K bytes for port JANWEBS to 199.3.217.200  
AD MSIR.0763: Sent      10K bytes for port JANWEBS to 199.3.217.200 in .001  
  seconds (9840K/sec)  
ST USERID='ALEX' ACCOUNT='ALEX' DKRD=2 DKWR=2 SQRD=2 SQWR=303 SGMTI=106 CPU  
  REQ=3 MOVE=10 AUDIT=11 WAIT=6 FINDS=4 SORTS=1 RECDs=3 DKAR=24 DKPR=288  
  PCPU=708 BXFIND=8 SCREENS=1  
AD M204.0352: IODEV=15, OK ALEX          ALEX          LOGOUT 08.135  MAY 14  
  07.58.17 -14
```



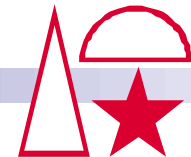
Saved Compilations and Group Recordsets (Finds, etc.)

- Size of variables in VTBL affected by number of files in a group
 - 8 bytes per file in a group for a Recordset, foundset or List
- Compilation can be saved or reused even if the group compiled against was a temporary group
 - So VTBL layout for compilation can vary from run to run
 - User running a request might require different VTBL layout than user that compiled request
 - If running user requires less VTBL for a recordset, the compiled recordset is simply not all used
 - But if compiled recordset is larger, the request must be re-compiled



Re-Compiling Requests Because of Group Differences

- Beware requests that might use multiple temporary groups
 - Re-compile forced for bigger group **A** might also have a smaller group **B**
 - So next user might re-compile, yet again because of a larger group **B**
 - So a request might get compiled over and over and over
- So, if possible, try to save compilation ahead of time with **all** groups at their maximum possible size



Conclusions

- Model 204's strategy for dealing with compilation best of all worlds
 - No permanent saved compilations eliminates issues managing saved compilations
 - But just-in-time pre-compilation of requests reduces cost of compilation to noise in most Onlines
- Worth understanding how it works to make sure you're getting full benefit
- SirFact adds extra capabilities to automatically re-compile request when a procedure in the request has changed
 - Not discussed here – out of time



Any Comments or Questions?



Sirius Software, Inc.