More Ways to Improve RACF Performance

By Robert S. Hansel and Robert L. Whittle

A previous article presented 10 ways to ensure RACF quickly and efficiently processes the thousands of logon and access authorization requests it receives each minute. This article, which is an update to our 2006 article, addresses 10 additional techniques for further improving performance, including the use of new features that have since been added to RACF. However, before we look at the new tips, let's revisit the original list (available at http://entsys.me/4pbeq):

1. Implement the Global Access Table (GAT)
2. Maximize in-storage resident data blocks
3. RACLIST classes whenever feasible
4. RACGLIST the RACLISTed classes
5. Increase Enqueue Residency (ERV)
6. Use Virtual Lookaside Facility (VLF) caching
7. Make efficient use of groups
8. Replace OPERATIONS with storage administration authorities
9. Implement Sysplex coupling facility caching
10. Cease gathering unnecessary statistics.

Now let's consider 10 new techniques:

1. Increase the sets of generic profiles stored in each address space. Started tasks, batch jobs, and Time Sharing Option (TSO) users frequently access multiple data sets with the same High Level Qualifier (HLQ) and multiple resources associated with the same general resource class. These data sets and resources are often protected by generic profiles. Generic profiles are RACF database entries that usually contain masking characters (e.g., *) and typically protect several data sets or resources with similar names. An example is PAY.MASTER.**, which would protect data sets whose names begin with PAY.MASTER.

Upon receiving the first request for access to a data set in a new HLQ or resource in a new class, RACF retrieves and stores a list of all the related generic profiles in the user's address space. This list is known as a Generic Anchor Table Entry (GATE). RACF uses these GATEs to identify the specific profile protecting each data set or resource the user attempts to access. After identifying the profile, RACF then retrieves and stores a copy of it in the user's address space for access authorization checking. RACF reuses these in-memory copies of profiles for subsequent access checking rather than repeatedly fetching the same profiles. Over time, a user address space will accumulate multiple GATEs and copies of profiles for rapid reuse.

In releases prior to z/OS 1.12, RACF keeps four GATEs in memory for each address space. Once all are in use, a request for a new HLQ or resource class causes RACF to discard the least recently referenced GATE, along with all its accumulated profiles, and replace it with a new GATE. An address space randomly accessing many different HLQs and resource classes may experience GATE thrashing, where lists and copies of profiles are constantly fetched, dropped, and fetched again. When the number of profiles involved is large, such address spaces may experience a noticeable degradation in performance.

In z/OS 1.12, RACF introduced a new option called GENERICANCHOR, which supports retention of more than the default four GATEs. Each address space may now have up to 99 GATEs. The number of GATEs can be specified either for the entire system, for individual jobs, or

a combination. The RACF operator command SET GENERICANCHOR is used to establish and modify these specifications. If you have a performance-sensitive address space that conceivably might be experiencing GATE thrashing, consider using this feature to increase its number of GATEs to match the number of HLQs and classes it might reference during normal processing.

If you RACLIST a general resource class as discussed in the previous article, RACF won't need to build GATEs for it because all the profiles are already stored in memory.

2. Periodically reorganize the RACF database. RACF retrieves data from its database in 4K blocks. Each block may contain one or more profiles or profile segments. Over time, RACF administrative actions can cause the following negative effects on performance:

- A profile can be expanded by commands such as PERMIT and CONNECT, causing it to spill over into an additional block.
- Profile and segment deletions can empty all but a small percentage of a block, wasting database and buffer space.
- Each newly added profile segment may be stored in a different block from its corresponding profile, thereby requiring additional I/O to fetch the segment. This most commonly affects logons involving a user profile and its TSO, CICS, or OMVS segment.

To address these issues, reorganize all RACF databases at least annually using the IRRUT400 utility. IRRUT400 realigns index and profile blocks into consecutive order, stores profile segments adjacent to their corresponding profiles, reclaims unused space, adds free space for subsequent growth, and eliminates any index errors.

3. Use GRS rather than hardware RESERVEs for sharing a RACF database. When two or more systems share a database in non-Sysplex, data-sharing mode (no coupling facility being used), RACF uses exclusive hardware RESERVEs to serialize the database for most updates. The system holding an exclusive RESERVE locks out all the other systems until it has processed all its update requests. This lock is on the entire DASD volume and affects all data sets on the volume.

Global Resource Serialization (GRS) can convert RESERVEs to global ENQs. Each system is given exclusive control for one update request at a time and the lock is only for the RACF database, not the entire DASD volume. Use of GRS avoids the contention and monopolization issues associated with exclusive RESERVEs.

4. Split the database into multiple data sets. A RACF database is usually allocated as a single pair of data sets, one for the primary and one for its corresponding backup. Such a RACF database has a single set of in-storage resident data block buffers. For large, highly active databases, these buffers may not be sufficient to adequately handle the workload during peak periods.

A z/OS RACF database can be split into as many as 99 primary/backup data set pairs, each with its own subset of profiles. The RACF range table ICHRRNG is used to specify how profiles are distributed across the various data sets. Each individual data set is given its own set of buffers.

The major drawback to splitting a RACF database is that a change to the ICHRRNG table requires a Sysplexwide IPL for implementation. Changes are needed occasionally to adjust how the profiles are distributed, especially if adoption of a new profile naming convention has created a profile placement imbalance. Scheduling such IPLs is becoming increasingly more difficult.

5. Isolate the RACF databases. Even if you implement many of the features mentioned in this pair of articles, your RACF database is still likely to incur significant amounts of I/O, especially during peak logon and system usage periods. Placing other high use data sets on the same DASD volume as the RACF database can delay the retrieval of profiles, potentially impacting the performance of all systems sharing the database. If possible, don't put any other data sets on a volume that contains a RACF database. If other data sets are placed on a volume with a RACF database, ensure they won't cause any hardware RESERVEs. This advice applies to both the primary and backup data sets, and, for a split database, to all component data sets.

6. Log judiciously. One essential function of security is monitoring system events to detect suspicious activity such as unwarranted use of high-powered authorities or attempts to access sensitive production files. Monitoring in a RACF environment creates System Management Facility (SMF) records. Excessive logging can potentially overwhelm SMF buffers. In addition, large amounts of DASD and tape resources can be required for record collection and archiving. While it's reasonable to log violations, access to sensitive data, and updates to system files, logging every data set access probably isn't. Take care when monitoring users who might access many z/OS UNIX files and directories as this can generate numerous SMF records.

This recommendation shouldn't be used as a justification for turning off the SETROPTS option OPERAUDIT, which monitors use of the powerful OPERATIONS authority. Installations that rely heavily on OPERATIONS authority should instead seek to replace its use with the storage administration authorities recommended in our prior article. The SMF data collected by OPERAUDIT is instrumental to identifying and eliminating OPERATIONS use.

7. Reduce updates to last-access date. Every time a user logs onto the system, RACF updates the "last-access" date and time in the user's profile, resulting in a write operation to the RACF database. RACF needs this information to enforce password change frequencies and perform automatic revokes due to inactivity. However, RACF only needs to know the most recent date that a user logged on. The most recent logon time is immaterial to these functions, and repeatedly updating this field throughout the day is of little value.

In z/OS 1.11, RACF introduced an option to limit updating of the last-access date to just the first logon of the day. This only occurs when an application (e.g., CICS) passes its APPLID to RACF, along with a USERID and password during the logon process and only when an APPL class profile protects that APPLID. When the APPLDATA field of the corresponding APPL class profile has the value "RACF-INITSTATS(DAILY)", RACF won't update the user profile when the user's last-access date has already been set to today's date.

By default, TSO doesn't pass an APPLID to RACF during log-on processing. Beginning with z/OS 1.10, it's now possible to instruct TSO to pass an APPLID and leverage this new performance feature. Instructions for activating this feature can be found in the *z/OS TSO/E Customization* manual.

8. Code NOYOURACC on RLIST commands. When a user issues an RLIST command to list the profile protecting a general resource, RLIST reports the user's own level of access to the resource. This seemingly innocent behavior can impact performance when the resource is protected by profiles in a member/grouping class pair. Because Universal Access (UACC) settings (i.e., default access) and access permissions in multiple profiles affect the result, RACF must retrieve and RACLIST all the profiles in the class pair, then perform an authorization check to determine the user's access. All this activity occurs in the user's address space. For member/grouping class pairs with thousands of profiles, a single RLIST may require significant I/O to the RACF database to fetch the profiles and large amounts of CPU time to process them. Usually, all this processing is wasted because the person executing the RLIST isn't interested in his or her own access.

RACF administrators with SPECIAL authority can bypass this processing by specifying the NOYOURACC parameter with the RLIST command. It can be abbreviated as NOY.

Use of RLIST parameters ALL and RESGROUP to list a member class resource will also result in all grouping profiles being retrieved and inspected to identify every profile covering the resource.

9. Avoid I/O-intensive commands and utilities during peak system activity. Certain commands and utilities can cause substantial amounts of I/O to the RACF database. Others lock the database and block all other use until they're finished. Commands LISTUSER * and LISTGRP *, for example, list every user and group, respectively, and each may need to read every user and group profile to gather the required information. The IRRUT200 database backup utility temporarily locks the database while it makes a copy. The IRRUT400 reorganization utility and the IRRDBU00 database unload utility also lock the database, although problems can be avoided simply by using a copy of the database for input rather than the live database. Executing SETROPTS REFRESH commands can generate a large number of I/O requests, especially when either a GENERIC REFRESH for data sets or a RACLIST REFRESH for a general resource class with many profiles is issued. Except for emergencies, delay execution of these commands and utilities until off-peak periods.

Note that CONNECT and REMOVE commands, which add and remove users from groups, each generate three updates to the database. Executing large batches of these commands during peak log-on periods can delay users trying to gain system entry.

We encourage every installation to take action to ensure that RACF responds to each request as expeditiously as possible.

10. Keep the database free of unnecessary profiles. Profiles take up space in the database as well as space in the database index, GATEs, and unload copies of the database, and they also consume CPU cycles when processed by commands and utilities. Profiles for non-existent users and resources and for unnecessary groups waste space and processing time. Their mere existence can confuse, mislead, and complicate research and analysis of system and security issues. Periodically review the validity of all profiles and remove those that are obsolete.

Conclusion

We often view RACF as the ultimate bureaucrat because everything must wait until RACF passes judgment. We encourage every installation to take action to ensure that RACF responds to each request as expeditiously as possible. This will entail a collaborative effort involving RACF administration, capacity planning, storage administration, and systems programming. We hope the suggestions in this and our prior article help you maximize RACF's responsiveness. ETJ

Robert S. Hansel is lead RACF consultant and president of RSH Consulting, Inc., a RACF professional services firm he founded in 1992. He has worked with RACF since 1986 as a manager, administrator, technician, analyst, auditor, and instructor. He supports several RACF users groups throughout the U.S.
Email: R.Hansel@rshconsulting.com; Website: www.rshconsulting.com

Robert L. Whittle is a senior RACF consultant at RSH Consulting, Inc. He has been a RACF systems programmer and consultant since 1992.
Email: R.Whittle@rshconsulting.com; Website: www.rshconsulting.com