Runtime Infrastructure - Bug #11743

Startup contention in TemporaryAccountPool/TemporaryAccountWorker stalls concurrent client logins.

08/13/2026 11:59 AM - Eduard Soltan

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eduard Soltan	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 08/13/2026 12:19 PM - Eduard Soltan

Every client spawn calls TemporaryAccount.open(), which acquires TemporaryAccountPool.taskLock, hands one task to the single TemporaryAccountWorker thread, and blocks on callerLatch.await() until that task completes. So N connection serialize behind N sequential worker task executions — no overlap.

What the worker does per task includes:

- HashPassword.hashPassword() at 210,000 PBKDF2-HMAC-SHA256 iterations.

- each create triggers two full rebuilds of the temp-account list (SecurityCache.refreshTempAccounts), once for the admin cache and once for the live cache.

A solution for HashPassword.hashPassword() slowness problem could look in the following way:

- Generate and hash the credentials on the posting thread, when the CreateAccountTask task is created. CreateAccountTask's creation-mode constructor does the random subject/password generation, the PBKDF2 hash, and the ordinal allocation, filling a ready-to-use TempUserDef. TemporaryAccountPool.createUser() constructs the task before entering synchronized (taskLock), so client threads hash in parallel instead of queueing to hash one at a time on the worker.
- Use a reduced iteration count for these ephemeral secrets. New HashPassword.EPHEMERAL_ITERATIONS (1,000) plus a hashPassword(String, int) overload. And will be used only for temporary users.

#3 - 08/14/2026 08:25 AM - Eduard Soltan

Every web login calls TemporaryAccount.open(), which reaches TemporaryAccountPool.createUser(). That method hands a CreateAccountTask to a **single** worker thread and blocks until it completes:

```
CreateAccountTask account = new CreateAccountTask();
synchronized (taskLock)
{
   poolTask.execute(account);
}
```

In trunk, CreateAccountTask.createUser() computes the account password hash (HashPassword.hashPassword(), PBKDF2, 210,000 iterations) **inside** that serialized section. Under load the generation of hash takes ~2 s, and every concurrent login queues behind it. I got this result by 70 concurrent logins on hotel.

I made some changes to generate the credentials and compute the hash in the CreateAccountTask constructor, on the calling thread, so the expensive work runs in parallel across all cores. Only the directory write and the security cache refresh remain inside the lock.

100 concurrent logins on hotel_gui. Timings from instrumentation around createUser().

variant	time taken to create 70 temporary accounts	avarage wait in lock
trunk (hash inside lock, 210k)	203.735 s	81,654 ms
11521a, 210k hash generation iterations	7.869 s	119 ms
11521a, 1,000 hash generation iterations	4.804 s	179 ms

As a separate thing I tried lowering the PBKDF2 iteration count (210000 -> 1000), because the password is generate at temoprary user creation and is deleted at the end of the seesion. But it does not contribute significantly to the the time drop in congested user login.

change	time taken to create 70 temporary accounts	share
hash moved out of the worker thread	203.735 s -> 7.869 s	98.4%
iterations 210,000 -> 1,000	7.869 s -> 4.804 s	1.6%

Measurement of creation of temporary user account shows a significat imporovment, but I still hit the same issue described #11521-194.

#4 - 08/14/2026 08:28 AM - Constantin Asofiei

Eduard Soltan wrote:

> Measurement of creation of temporary user account shows a significat imporovment, but I still hit the same issue described #11521-194.

For #11521-194 are we talking about a process-storm and thus the system is under heavy load and can't execute all in time, or again lock contention?

Please add any logging from Serban to this branch.

#5 - 08/14/2026 08:30 AM - Șerban Bursuc

- File HashPassword.patch added

Eduard, I used this patch over my load test branch. Then ask an LLM to aggregate the data, I do not have a Python script to do this.

#6 - 08/14/2026 08:31 AM - Șerban Bursuc

- File tempaccount-instrumented.patch added

Wrong patch, sorry, it's this one.

#7 - 08/14/2026 10:57 AM - Eduard Soltan

- Status changed from New to WIP

- Assignee set to Eduard Soltan

I think it's a process storm. Not lock contention. When many logins arrive at once, all of those JVMs start at the same moment and compete for the same CPUs. And they do finish toghther, so I think it is not a lock contention.

Measured on a 16-core machine:

logins at once	time each
1	2.0s
5	10.4s
10	23.6s

Committed the changes on 11743a, rev. 16704.

#8 - 08/14/2026 10:58 AM - Eduard Soltan

- % Done changed from 0 to 100

- reviewer Constantin Asofiei added

#9 - 08/19/2026 07:25 AM - Greg Shah

- Status changed from WIP to Review

#10 - 08/21/2026 07:31 AM - Constantin Asofiei

- Status changed from Review to Internal Test

I'm OK with the changes.

#11 - 08/21/2026 07:34 AM - Eduard Soltan

Constantin Asofiei wrote:

> I'm OK with the changes.

I could test ETF and clustered application, anything else should be tested?

#12 - 08/21/2026 07:38 AM - Constantin Asofiei

Eduard Soltan wrote:

> Constantin Asofiei wrote:
>
> > I'm OK with the changes.
>
> I could test ETF and clustered application, anything else should be tested?

This affects agent spawning (or other load-testing). It does not affect runtime or anything else. Test something with classic agents and is enough.

#13 - 08/21/2026 08:50 AM - Eduard Soltan

Constantin Asofiei wrote:

> This affects agent spawning (or other load-testing). It does not affect runtime or anything else. Test something with classic agents and is enough.

Tested web_api project with a clasic appserver, plus large gui application was tested on load with 300 concurrent users.

#14 - 08/21/2026 09:05 AM - Constantin Asofiei

- Status changed from Internal Test to Merge Pending

Can be merged after 11592a. Thanks.

#15 - 08/21/2026 10:34 AM - Eduard Soltan

- Status changed from Merge Pending to Test

11743a was merged into trunk rev. 16718 and archived.

Files

HashPassword.patch	6.16 KB	08/14/2026	Şerban Bursuc
tempaccount-instrumented.patch	18.8 KB	08/14/2026	Şerban Bursuc