imperva

WHITE PAPER

Overhead of Database Native Auditing in the Real World

Auditing-related overhead imposed on databases is one of the most misunderstood topics in the field of database security, both when using native capabilities as well as when using DAM agents. Native auditing subsystems have evolved quite a bit over the past 10 years, to the point where the performance impact on the databases for data collection is on par with agent-based collection. Since monitoring/auditing overhead differs not only in the collection method but more so on the auditing policy and the amount of data collected, any statement that asserts overhead will be X% is misleading and merely a way to dismiss a frank review.

This whitepaper provides summary empirical results of overhead using native methods based on real-world scenarios. It summarizes observed results for the most common cases and then provides a discussion on how these results can be used to extrapolate to additional scenarios. Since an understanding of "how things work" will help you better understand the numerical results put forth in this whitepaper, we strongly recommend that you read Imperva's "Understanding the Overhead of Database Monitoring & Auditing" prior to reading this as it provides a thorough discussion of the internals.

The numbers presented are consistent across multiple database platforms (both RDBMSs and NoSQL) so long as the version of the database is from approximately the last decade. Very old database versions have in the past logged audit data in a synchronous manner and therefore exhibit higher overhead. However, even in these very old versions a realistic audit policy usually implies audit events that are still under 10% overhead (whereas modern databases, as shown below, are well under 5%).

Finally, measurements were also made using DAM agents in the lab. Agent overhead was found to be similar to using agent-less techniques in all practical scenarios, including scenarios that cannot be implemented using connection-based approaches such as monitoring access to sensitive data or monitoring based on privileges and admin commands.

"The granular nature and precise control supported by audit policies in modern databases ensure that the overhead measured both as CPU overhead and, more importantly, as impact to workloads and response times remain low (under 5%). This is true across database platforms, workload types, host loads, and scenarios."

- Ron Bennatan, SVP & GM, Data Security, Imperva.

1. Description of measured scenarios

The presented scenarios are based on real-world audit and security use cases. While dozens of scenarios were measured, they all fall into two activity-level scenarios (along with combinations of these).

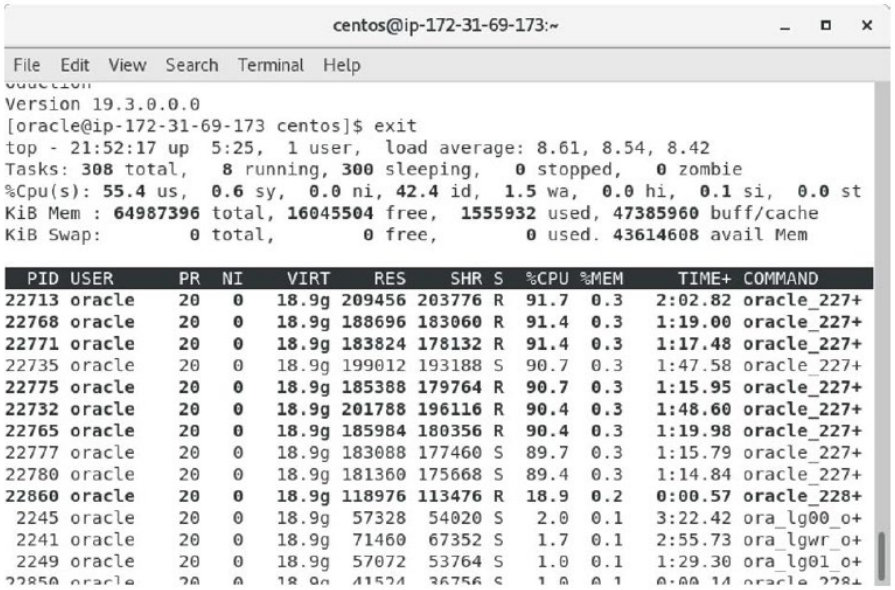
Connection-level scenarios mean that some connections are fully audited, and others are not audited at all. An example is privileged user monitoring using connection attributes such as the username, the program name, the client IP, or any such tuple. Conversely, the policy may be based on excluding certain connection tuples representing the application servers. The effect in terms of performance overhead is the same since what matters is how many log records are produced. Evaluation (the "if" part) is done upon connection.

Activity-level scenarios mean that the determination of what to audit can only be made based on the SQL, or what is actually being performed. This includes sensitive-data policies where access to certain tables/objects needs to be logged, where usage of certain commands need to be logged (including for example security commands like create user or export commands that are performed no matter by whom), or where certain privileges are used no matter what the connection tuple is.

In these cases, multiple results were aggregated differing in both workload type as well as what percentage of the activities need to be logged.

2. Overhead results

It's important to define exactly what "overhead" means. The easiest thing to do is to look at the CPU utilization and measure how much harder the machine is working. That is what most logging and auditing benchmarks do and what we originally did because it is the easiest thing to measure. As an example, Figure 1 shows Oracle processes running a heavy workload together with a medium access-level monitoring policy. The main Oracle processes (these starting with oracle_227) are serving the applications and consuming full cores (i.e., they are efficient). The three processes at the bottom, ora_lg** are the ones doing the auditing and together remain below 5%. Note that this is not 5% of the machine – i.e., the 2.0% used by ora_lg00 is 2% of a single core. Heavier audit policies will require more resources but not even coming close to what the application workloads are. Also, note that there is practically no system time used (55.4% user and 0.6% system). This is contrary to scenarios using DAM agents that consume system resources that affect everything running on the system.



But this is not exactly what matters most. We are trying to determine the impact on systems when adding monitoring. Therefore, the results summarized below measure not a CPU utilization but rather the relative slowdown in terms of looking at a complete workload from start to finish that simulates a complex application, applying different policies while the workload is running, and observing how long it takes for the entire workload to complete. This is the only way to truly gauge the impact of monitoring.

It was especially surprising to see how different impact on workloads is vs. CPU utilization when measuring DAM agents. Because DAM agents also involve kernel modules, much of the overhead of auditing is reported as system time and looking just at CPU consumption by processes is misleading. We often saw cases where the CPU of the agents was showing X% (e.g., 5%) but the impact on workloads finishing was 2X and sometimes even 3X (e.g., 10% - 15%).

The results below are based on a complex workload that includes application connections and "privileged" / "interactive" user sessions. The majority of queries are performed by the app sessions and overhead is measured by how much longer all the app sessions take to fully complete the work. Specifically, between 1% and 3% of all work is done by the admin/privileges sessions that are fully logged. Within the app connections (the app workload), some commands are logged and other are not. This granular monitoring – for example, access to sensitive tables, performing certain activities such as exports, or using elevated privileges (that can be hijacked) is not possible with DAM agents. Four cases were measured - logging 10% of all activity, 20% of all activity, 50% of all activity and 100% of all activity. Finally, because some database queries are faster than others (e.g., updates usually take longer than index-based selects) two different profiles were done – one with many updates and one with almost only SELECTs. Since overhead is measured in percentage, when queries perform very quickly the overhead is larger even though the audit time takes exactly the same time and resources (this is another reason why measuring impact is better than measuring CPU time).

Finally, the measurements were taken on heavily loaded servers (see Figure 1). This is important since a lightly-loaded box or a big box with a workload that does not stretch the server will often have even lower overhead because writing audit records is done asynchronously and therefore if there is no contention on resources, overhead

POLICY TYPE	WORKLOAD TYPE	PERCENTAGE OF ACTIVITY LOGGED	OVERHEAD	#
MONITORING ADMIN SESSIONS ONLY	ALL	ALL	0-1%	1
MONITORING ALL SESSIONS	MIXED WORKLOAD	LOGGING 5% OF QUERIES	0.5-1%	2
		LOGGING 10% OF QUERIES	1-1.5%	3
		LOGGING 20% OF QUERIES	1.5-2.%	4
	FAST QUERIES ONLY	LOGGING 5% OF QUERIES	1%	5
		LOGGING 10% OF QUERIES	1.5-2%	6
		LOGGING 20% OF QUERIES	2-3%	7
MONITOR AND LOG ALL ACTIVITY (100% LOGGING)	MIXED WORKLOAD	LOGGING 100% OF QUERIES	2-3%	8
	FAST QUERIES ONLY		3-4.5%	9

3. Example Audit Scenarios

Section 4 of the "Understanding the Overhead of Database Monitoring & Auditing" whitepaper presents the common requirements for monitoring activity in a database:

1. (Only) privileged user monitoring.
2. Privileged activities monitoring.
3. Sensitive data access monitoring.
4. Monitoring changes to critical data.
5. Full monitoring and logging.

For each of these scenarios the observed overhead is small, specifically:

3.1 (Only) Privileged-user Monitoring

Privileged-user monitoring involves an audit policy that audits all activities performed by DBAs, security admin, as well as activities performed using fire IDs and some service accounts. These audit policies can use fixed lists of which users/ connections to monitor or, conversely, could involve a list of users/connections not to monitor – i.e., rather than say which users to monitor, one defines which connections to omit (e.g., application server connections). This is the simplest possible audit policy, usually has the lowest overhead, and matches row #1. Imperva architecture enables movement towards a more open policy without the risk of blowing up appliances based on the leveraging of Big Data technologies.

3.2 Privileged-activities Monitoring

Privileged-activities monitoring is a refinement of 3.1 that prevents bypass and complex list management. The problem with 3.1 is that when fixed lists are created, they can easily be bypassed, and they need to be constantly updated. If a privileged user knows that a certain app account is not being monitored, they can log in using that account and fly under the radar. If a fixed list of privileged users is used to decide what connections to monitor, one can create a new user that will be used to perform malicious activity and no one will be the wiser. Since the compliance requirements are actually about administrative activities and privileges, it is sometimes better to base the audit on these activities and not on the connection attributes. Thus, if an app session starts performing problematic commands such as creating users, it will not be ignored.

With an agentless approach, the overhead is still row #1. Using agent-based interception to implement this scenario means that all activities bar none need to be copied and sent to the appliance for analysis. This is because the agent cannot look into the SQL – the SQL is only analyzed by the code running on the appliance. This scheme implies high overload on the host, agent and DAM appliance.

3.3 Sensitive Data Access Monitoring

Sensitive data monitoring is second only to monitoring elevated privileges. Sensitive data monitoring is common in privacy regulations, PII/ NPI-based requirements, PCI and more. It differs from 3.1 in that it requires broad monitoring/ auditing when anyone accesses sensitive data. It is also needed in order to create baselines and profiles even for app access so that AI and machine learning can be used to discover issues in excessive access to sensitive data.

Similar to 3.2, agent-based interception means that all data needs to be copied, sent, and analyzed by the DAM appliance - thus high overhead is observed. The reason is the same – the DAM agent has no awareness of what object is being used and therefore all SQL needs to be shipped to the DAM appliance. Here too, native auditing provides selective control to audit all actions and even specific actions on specific objects (such as READ vs WRITE).

Overhead in the native approach is shown in rows 5-7 depending on the workload type and the percentage of activity against the sensitive tables.

3.4 Monitoring Changes To Critical Data

Overhead is one of rows 2-4 depending on the percentage of activity against the sensitive tables.

3.5 Monitor And Log Everything

It may surprise many that have done database audit benchmarks decades ago that even policies that log everything have very little impact on the application performance. Results are shown in row 8 and 9. In fact, on some systems (for example, modern versions of Oracle) we even observed as little and 2.5% overhead even when logging everything and even when queries were mostly SELECTs.

4. Summary

The granular nature and precise control supported by audit policies in modern databases ensure that the overhead measured both as CPU overhead and, more importantly, as impact to workloads and response times remain low (under 5%). This is true across database platforms, workload types, host loads, and scenarios. This is also common across both on-prem and cloud database as a service.