

Additional Performance Tips

- [What is performance?](#)
- [What kind of performance is normal? How do I know if I have a problem?](#)
 - [Individual page display](#)
 - [RDF loading](#)
 - [Inference recomputation and search index rebuilding](#)
- [Tools for measuring performance](#)
 - [Testing without local modifications](#)
- [Tuning for improved performance](#)
 - [Memory](#)
 - [Server connections](#)
 - [MySQL configuration](#)
 - [In-memory temporary tables](#)
 - [Key buffer size](#)
 - [InnoDB buffer pool size](#)
 - [Transaction logging](#)
 - [HTTP caching](#)
 - [Alternative triple stores](#)
 - [Misbehaving robots](#)

What is performance?

Performance can mean different things to different sites including the length of time it takes to render a large page (e.g., a person with 800 - 1500 publications), to display a visualization, to load new data, to regenerate the search index or recompute inferences, or to generate an export of RDF data.

What kind of performance is normal? How do I know if I have a problem?

This section gives some very rough guidelines for determining whether your VIVO is performing similarly to established production installations on typical modern server hardware or virtual machines. The numbers below assume that VIVO is otherwise idle; that is, not loaded with concurrent public page requests or performing other background operations.

Individual page display

The time it takes to render an individual page can vary significantly depending on the types of data involved. The page for a person with many publication citations will take longer to render than one with simple links to other individuals. As a very general rule, your VIVO should be able to handle around 100 data items (properties) per second when displaying an individual page. Thus, if the page for a person with 500 publication links displays in five seconds, there may be relatively little room for performance tweaking short of caching the entire page. If the page takes 50 seconds to appear, there is very likely a serious performance bottleneck somewhere in the installation or a hardware deficiency that needs to be addressed.

RDF loading

Loading RDF through VIVO is slower than inserting it directly into the triple store because VIVO performs additional operations such as inference and search index maintenance as the data are changed. You should still expect to see at least several hundred triple insertions per second.

Inference recomputation and search index rebuilding

These operations are important for VIVO installations that modify data directly in the triple store instead of adding or removing RDF through VIVO. You should expect inference recomputation to average about 20-25 milliseconds per individual. (You can find your values in `vivo.all.log`.) Search index rebuilding is typically faster, on the order of 10 ms per individual.

Tools for measuring performance

Members of the VIVO community have found the following tools helpful in testing and measuring a site's performance:

- Google Analytics. Records some basic performance metrics in the Behavior > Site Speed section, such as average page load time.
- JMeter. Generates simultaneous connections for testing of performance under real-world production loads.
- New Relic. Software analytics suite including JVM and MySQL monitoring.

Testing without local modifications

Local code modifications – especially custom list views and filter policies – can introduce inefficiencies that lead to poor performance. Similarly, code under development may contain performance regressions or new features that have not yet been optimized. If you have made any such modifications or are using pre-release code, it is important to test performance when your VIVO database is used with an official VIVO release. If the observed performance differs significantly from that exhibited by a modified version, the modifications are suspect.

Tuning for improved performance

Memory

Ensure that that Java JVM for your VIVO has been allocated sufficient memory (heap space). This is a critical element of the installation process, as the default Java heap setting will cause VIVO to run extremely slowly. A production VIVO installation should typically be allocated several gigabytes of heap space.

Additionally, ensure that your server has enough memory to support the heap space you have allocated. Otherwise, data may be swapped to disk, which can seriously degrade performance. On a server that runs only VIVO, the available memory should be about double the Java heap space.

Server connections

A production VIVO installation often involves an Apache web server, the Tomcat servlet container, and a MySQL database server. The numbers of available connections between each of these servers should be set to prevent unnecessary bottlenecks. Thus, the maximum number of database connections should slightly exceed the number of possible concurrent Tomcat threads, which should in turn exceed the number of simultaneous Apache worker threads or child processes.

MySQL configuration

Data display in VIVO often depends on complex SPARQL queries that, when using the default SDB triple store, are translated into similarly complex SQL queries. Tuning the MySQL database server can significantly increase performance. There are a number of tools available for assisting with this process, such as [mysqltuner.pl](https://github.com/rackerhacker/MySQLTuner-perl) (<https://github.com/rackerhacker/MySQLTuner-perl>). There are also a few typical parameters that often require adjustment.

In-memory temporary tables

The nature of the SQL queries generated by the triple store often requires the generation of temporary tables. Ideally these temporary tables will remain in memory; if they exceed the threshold where MySQL writes them to disk, this can result in serious slowdowns. Depending on the amount of data in your VIVO and your server's available memory, you may need to increase the size limit for in-memory temporary tables.

Consult the MySQL documentation for the parameters

- `tmp_table_size`
- `max_heap_table_size`

Key buffer size

If your VIVO database uses MySQL's traditional MyISAM storage engine, consult the documentation for the `key_buffer_size` parameter. Increasing this value can yield significant performance benefit.

InnoDB buffer pool size

If your VIVO database uses MySQL's newer InnoDB storage engine, consult the documentation for the `innodb_buffer_pool_size` parameter. Setting this value as large as possible given available memory will improve performance.

Transaction logging

Changing MySQL's transaction logging settings can lead to dramatic improvements to the speed at which triples are added to or removed from the database. For more details, see „Writing the MySQL transaction log" here: [MySQL tuning, and troubleshooting](#)

HTTP caching

If VIVO's dynamically-generated pages do not exhibit acceptable load times, you may wish to enable HTTP caching. See [Use HTTP caching to improve performance](#). With this configuration, subsequent requests for pages whose contents have not changed will result in those pages being served directly from a cache instead of being regenerated from data in the triple store.

Alternative triple stores

While VIVO is tested with and configured by default to use Jena's SDB triple store with the MySQL database server, VIVO also includes support for TDB and Virtuoso as well as the ability to connect via HTTP to a SPARQL 1.1-complaint endpoint. Use of a different store may yield performance improvements, offer additional possibilities for performance tuning, or enable features such as clustering and load balancing. In addition, configuring SDB to use a database server other than MySQL may offer advantages for your installation. Note that some of the SPARQL queries in the [list views](#) employed by VIVO in page rendering have been optimized for SDB/MySQL with substitution of UNION for OPTIONAL. These queries should be modified for optimum performance with other stores that do not exhibit the same quirks.

Misbehaving robots

In some cases, poor VIVO performance has been traced to search engine robots that either ignore or misread directives in VIVO's robots.txt file, or which issue requests for large pages at a rate that greatly exceeds the demand otherwise encountered in typical production use. If the search engine in question is not critical to VIVO's visibility, it may be advisable to restrict access to the associated robots. In some situations, institutional search appliances are responsible for the excessive server load. Here, discussions with local IT staff may be warranted.