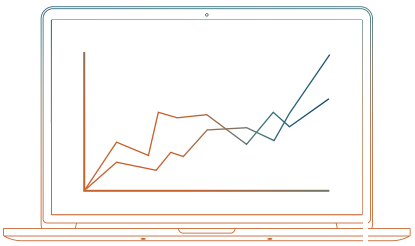
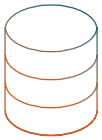
Tableau Best Practices
Performance Checklist

interworks

This performance checklist is based on recommendations from InterWorks' server architects for Tableau Server architecture and dashboard design best practices. Additionally, InterWorks created the Tableau Performance Analyzer, which can be used to identify opportunities for improvement in accordance with these best practices. Learn more at: www.powertoolsfortableau.com



DATA

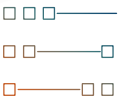


- [ ] Keep analysis simple. Work with a subset of your data. Extract a sample if needed.
- [ ] Bring in only the data needed for analysis. Consider adding a data source filter or using an extract. If using a join, minimize the number of joined tables.
- [ ] Use "Describe" to explore dimensions in new data sets without having to load them into a viz. (Keyboard shortcut CTRL+E)
- [ ] Remove unused columns (measures/dimensions) in order to minimize extract refresh time or custom SQL query time.
- [ ] Create a published TDS file for your business team to use instead of each Tableau Desktop user creating and publishing their own data source. This includes all metadata associated with dimensions, measures, calculated fields, hierarchies, sets, parameters and naming conventions.
- [ ] Use extracts wherever possible to accelerate performance. Hide unused and confidential fields. Roll up data granularity by pre-aggregating or filtering. Break hierarchies to only visible dimensions.

CUSTOM SQL

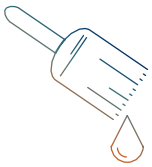
- [ ] Limit custom SQL connections as they can be inefficient. Where possible, create a view on the database server to implement your custom SQL and connect Tableau to your view.
- [ ] Avoid parameters in custom SQL in Tableau. Tableau wraps the custom SQL in a subquery that many databases don't handle well. Consider building a view in the database or use a multi-table join with filters.
- [ ] Watch for useless clauses, e.g. ORDER BY. Tableau is going to re-sort the data once loaded anyway.
- [ ] Avoid pre-aggregating in custom SQL.

CALCULATIONS



- [ ] Use calculated fields carefully. Think about the data type as you code the calculation. Number and Boolean > date > string calculations when it comes to performance.
- [ ] Limit blended calculations. They require sequentially querying multiple data sources and can be time-consuming. Where possible, create a view on the database server.
- [ ] Avoid row-level calculations involving parameters. This is an expensive operation.

RENDERING



- [ ] Avoid high mark counts. More marks = longer rendering time.
- [ ] Limit the use of detailed text tables with lots of marks.
- [ ] Minimize the file size of any images or custom shapes where possible. As a general rule of thumb, keep images under 50kb.
- [ ] If using custom shapes, use transparent background PNGs instead of JPGs. Views will render cleaner, and shape files will take up less space.

LOCAL COMPUTATIONS

- [ ] Even if a workbook is published to Tableau Server, local computations still impact performance. 85% of the computation and rendering happens on Tableau Server. Leverage the power of Tableau Server whenever possible by limiting local computations such as groups, hierarchies, reference lines, table calculations, and blending.
- [ ] Table calculations are powerful, but they can be slow. They are dependent on the local computation engine and can require substantial memory. VizEngine in v8.0 helps with this.
- [ ] Data blending builds a secondary temp table in cache. Although pre-aggregated, it is still computed locally. In v9.0, Tableau will begin processing queries in parallel, but it will be dependent on the data source. Until 9.0 releases, all queries run in series (sequentially).

FILTERING

- [ ] Minimize the number of quick filters. Use dashboard filter actions where possible.
- [ ] Avoid selecting “Only Relevant Values” for your quick filters. This requires sequential queries. Do not use this when not needed.
- [ ] Avoid high-cardinality quick filters (multi-select or drop-down lists). High-cardinality quick filters are slow to load and render.
- [ ] Avoid quick filters or actions that drive context filters. These require reloading the context table and should be avoided wherever possible.
- [ ] Keep range quick filters simple. The more complex the range, the slower the query.
- [ ] Replace quick filters showing “Only Relevant Values” and high count of quick filters with dashboard filter actions. They will cascade as your user interacts, and they perform faster.
- [ ] Don’t be lazy with user filters. Security by user filters can impact performance on Tableau Server as the server cannot share connections and query caches if user filters are active. Consider building a summary view that is a user-agnostic overview using a pre-aggregated extract with underlying data hidden. For a detailed view, restrict it to specific users or active directory groups instead of user filters.

DASHBOARD LAYOUT

- [ ] Limit the number of worksheets on a dashboard. If you have more than four visualizations on a dashboard, strongly reconsider.
- [ ] Fix dashboard size relative to end-user consumption. Automatic sizing is less efficient than specifying dashboard size.