VIVO v1.8.1 Release Notes

What is this document?

The VIVO 1.8.1 release concentrates on fixes for certain bugs and performance issues. Some minor, non-breaking additions are present in the ontology, and a non-breaking addition to the UI.

Performance Improvements

	Release 1.7	Release 1.8	Release 1.8.1 RC
Small data set (800,000 triples, 200 people, 3,500 articles)	100%	292%	**70%**
Large data set (11,000,000 triples, 4,500 people, 40,000 articles)	100%	256%	**72%**

Relative execution time. Testing by Jim Blake, as reported on the development list using the 30 slowest profiles.

Note that all testing reported below has been performed on a Macbook Pro, with PCI-E SSD. However, no specific tuning has been applied to either the hardware or software. Real world performance will depend on hardware and software configuration - it is recommended that you have an SSD / high IO performance storage layer, and if using SDB/MySQL, enough memory allocated to read the tables indexes.

Page Rendering

AntiSamy is no longer used to filter fields before they are rendered. For a large profile in a test dataset, this was responsible for over two seconds of the execution time required to render a profile.

A simple regular expression is used to filter out and JavaScript elements - this is 300x faster than using AntiSamy.

- Most pages - even large profiles - render within approximately two seconds
  - Large profile in a test dataset takes between 1.5 and 2 seconds to render (It has been reported that the same large profiles took 4.3 seconds in v1.7, and 5.3 seconds in v1.8)
  - Large profile when logged in as root user takes 6.5 seconds to render (was reported to be 7.75 seconds in v1.7, and 14.7 seconds in v1.8)
  - Large profile when logged in as site admin user takes 2.5 seconds to render
- Worst case profile tested - 1500 publications, high number of co-authors, between 3.5 and 4.5 seconds
- Manage publications / grants organisation pages performance improvements
- Manage people in organisations page now include the position label with each person entry so that you can disambiguate multiple person entries

Performance Tip: Site Admin by default does not display the "related by" faux property - this is responsible for the majority of the performance hit when logged in as root

Visualizations

All visualisations have been overhauled

- Map of Science and Temporal Graphs significantly faster
    - Under 3 seconds for a 1.2 million Quad dataset (previously 1 minute 20 seconds)
    - Approx 2 minutes for a 24.6 million Quad dataset. Contains 145,000 people, 155,000 publications and 14,000 journals
  - Person level Map of Science return in under 2 seconds, using direct queries of the triple store
  - Person level Map of Science will use the system-level cache once queries take longer than 2 seconds, if the system-cache has been populated
  - Background refresh of Map of Science / Temporal Graph data - once populated, all requests are served from the cache whilst refreshes occur
- CoAuthor and CoInvestigator visualisations use short-lived caches to prevent multiple executions of the same query in rendering a single visualisation
- Minor tweak to CoAuthor query to improve performance
- Sparklines use some of the under the hood improvements
- **New** Added AltMetric embed code to display badges on the article pages - enabled by default, disable via the runtime.properties
- **New** D3 based versions of Co-Authorship and Co-Investigator available - switch between D3 and Flash via the runtime.properties

Enabling The New Visualizations

Both the AltMetric badges and D3 visualisations are enabled by default, but can be disabled by the presence of the appropriate settings in the runtime.properties. If you are upgrading an existing installation, and do not adjust your runtime.properties, then you will start to see the new visualizations.

If you do not want to have AltMetric badges then add the following to your runtime.properties:

```
resource.altmetric=disabled
```

If you do not want to have the D3 visualisations, and wish to have the Flash based Co-Authorship and Co-Investigator visualisations instead, add the following to your runtime.properties.

```
visualization.d3 = disabled
```

For more information on the new settings, please see the example.runtime.properties file.

Click here to see an example of the new Visualizations in 1.8.1

Memory Usage

- New data structures for Map of Science / Temporal Graphs use lightweight Java objects instead of Jena models (should use much less memory)
- Search Indexer does not queue statements to index if paused and a full rebuild has been requested (much lower memory usage during reasoning)

Under The Hood

- Reduce overhead on reinferencing - up to 30% faster on individuals with no changes (depending on configuration)
- Reasoning on a large dataset has more consistent performance (used to slow down / crash with memory used by the search indexer)
- Faux property resolution rewritten to greatly reduce work being repeated in the presence of multiple instances of the same property
- RDFService has additional methods
    - CONSTRUCT that takes a Model to write into
    - SELECT that takes a ResultSetConsumer - implemented by the user - which processes each QuerySolution as it is retrieved from the ResultSet
    - Reduce latency and memory overhead of reading into a Jena model; serialising; and then re-reading into a Jena model in the calling method.
      (NB: Responsible for 20 seconds of improvement to Map of Science / Temporal Graph)
- Replace certain uses of RDFServiceDatasetGraph with RDFService (repeated calls to find() in RDFServiceDatasetGraph responsible for some overhead)
- RDFServiceSDB always constructs queries against the graph, and not the union model (simple optional queries much faster against the graph than the dataset)
- Clean up of many list view SPARQL queries, removing a few redundant patterns.
- Cache list of graph lists when using a SPARQL backend for faster page loads (4 second saving on Virtuoso / 25 million triples)
- **NOTE**: Some methods have changed their signatures to support the above. If you have custom Java code in your installation, you may need to make minor adjustments - typically, this will be exchanging a Dataset parameter for an RDFService.
- **NOTE**: Some listview-*.xml files have changed, if you have customised your list views, you will need to resolve the conflicts.
- **NOTE** List views that return publications (e.g. authorInAuthorship) now only resolve the editor person for publications that are either bibo:Book or bibo:BookSection (includes Chapter, etc.). This is necessary for reasonable performance when you have large publication lists that involve articles with many co-authors.

Bug Fixes

- Pause counting on the search indexer to prevent it become accidentally unpaused during long running processes (e.g. reasoning)
- VIVO-1059 Improved parameter binding in SparqlQueryDataGetter
- VIVO-1075 Correct use of Jena Nodes to access typed data (MarkLogic)
- VIVO-1046 vCard authors do not display if lacking first name
- VIVO-1047 vCard middle names displayed before first names
- VIVO-1038 vCard grant contributor behaves as publication author
- VIVO-1081 Fix to display of training positions within an organisation entry
- VIVO-1114 Broken sparklines when more than 1000 publications, etc
- VIVO-1122 Fix to resource usage for running on SPARQL triple stores

Additional Changes

- TinyMCE filters out Word formatting on paste
- TinyMCE version updated
- Add seven US provinces to us_states.rdf
- DOI property displays as a link