Versioning Performance

OnParentVersion with many descendants

All tests involved creating a single container resource with one or more descendant resources. Versioning had three modes: enabled on all test containers (TRUE), disabled on all test containers (FALSE), or enabled on the root container and on half of the children (HALF). onParentVersion was set to either VERSION or COPY (for non-versioned tests this has no effect). Scaling was tested by increasing the number of descendants (100, 1000) and by increasing the size of the jcr:content property on the descendant (roughly 10kb vs 1mb). Three types of descendant resources were tested: auto-named children resources with auto-generated intermediate folders (child), named children resources (named), and binaries (DS).

After all the descendants were created and assigned the versionable mixin, when appropriate, a single new version of the root resource was created (New Version Time). Total time spent creating descendants and enabling versioning on them is recorded as Total Child Creation Time.

Time to create a new version of the root resource

The creation of a new version of the root resource was only a small portion of the overall time spent in setting up each test (~1.6% mean, 7.6% max), although there was a significant increase in the amount of time required when auto-named children were used as the descendants. Generally, using COPY mode resulted in a slower new version creation times than VERSION.

Descendant Type	# of Descendants	Binary size	COPY (ms)	VERSION (ms)
child	**100**	**10,000**	572	616
		1,000,000	1,119	956
	1000	**10,000**	7,227	7,463
		1,000,000	7,600	7,009
DS	**100**	**10,000**	172	75
		1,000,000	222	117
	1000	**10,000**	1,369	262
		1,000,000	1,925	288
named	**100**	**10,000**	432	54
		1,000,000	438	99
	1000	**10,000**	5,471	247
		1,000,000	4,986	290

Disk Usage

Results for disk usage were not always consistent, particularly for small batches where in some cases the size of the binary store actually decreased after a test, possibly due to background cleanup processes occurring in Modeshape. Usage was measured by using the du command on the data directory before and after running the test. Binary content was never duplicated in any of the descendant types or onParentVersion modes tested. For descendants with larger binary content the increase in disk usage for using any type of versioning over not versioning was small.

Descendant Type	Number of descendants	Binary size	Not Versioned	COPY (ms)	VERSION (ms)
child	100	10,000	2,880	7,259	7,808
		1,000,000	132,636	136,706	136,629
	1000	10,000	37,094	77,579	77,143
		1,000,000	1,325,440	1,368,764	1,370,260
DS	100	10,000	1,678	3,506	2,658
		1,000,000	102,832	105,066	104,563
	1000	10,000	16,443	33,852	31,444
		1,000,000	1,315,988	1,331,370	1,324,400

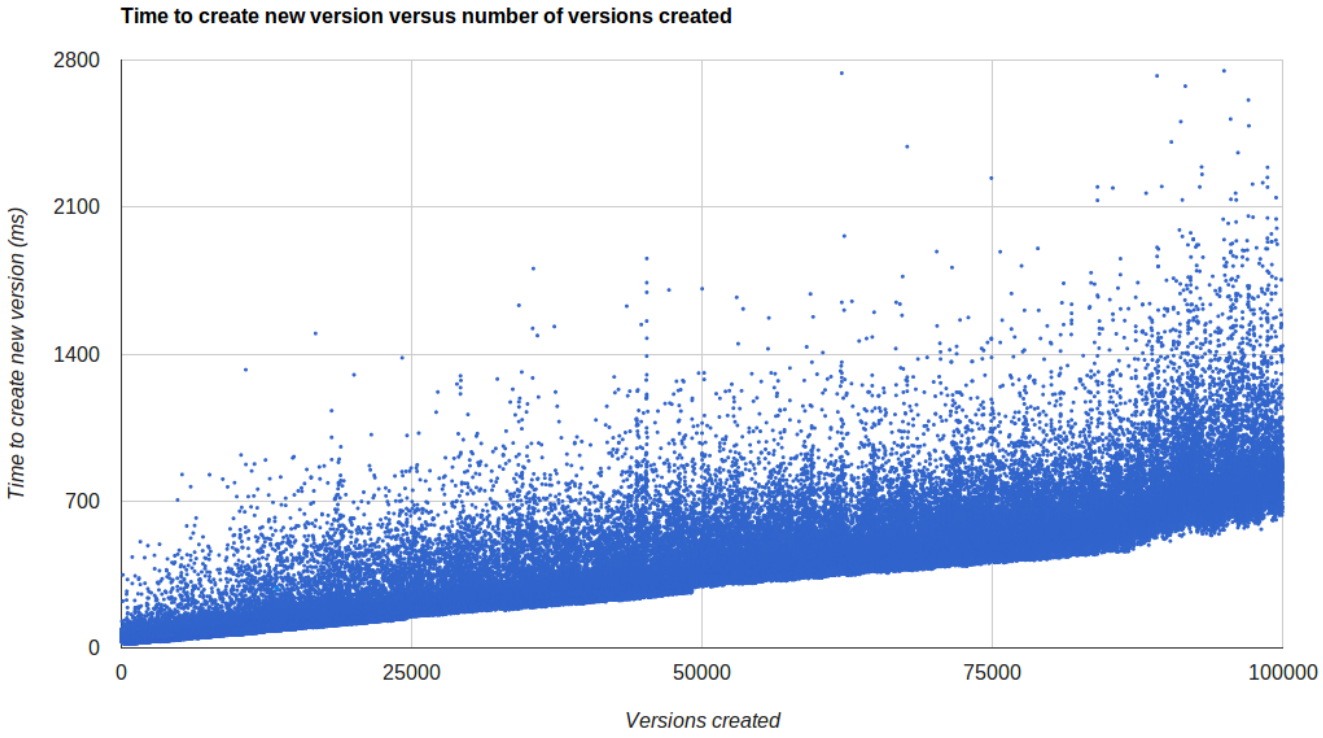
named	100	10,000	3,088	6,360	4,472
		1,000,000	132,012	135,294	133,525
	1000	10,000	33,792	63,904	47,862
		1,000,000	1,323,540	1,353,080	1,336,844

Full data results:

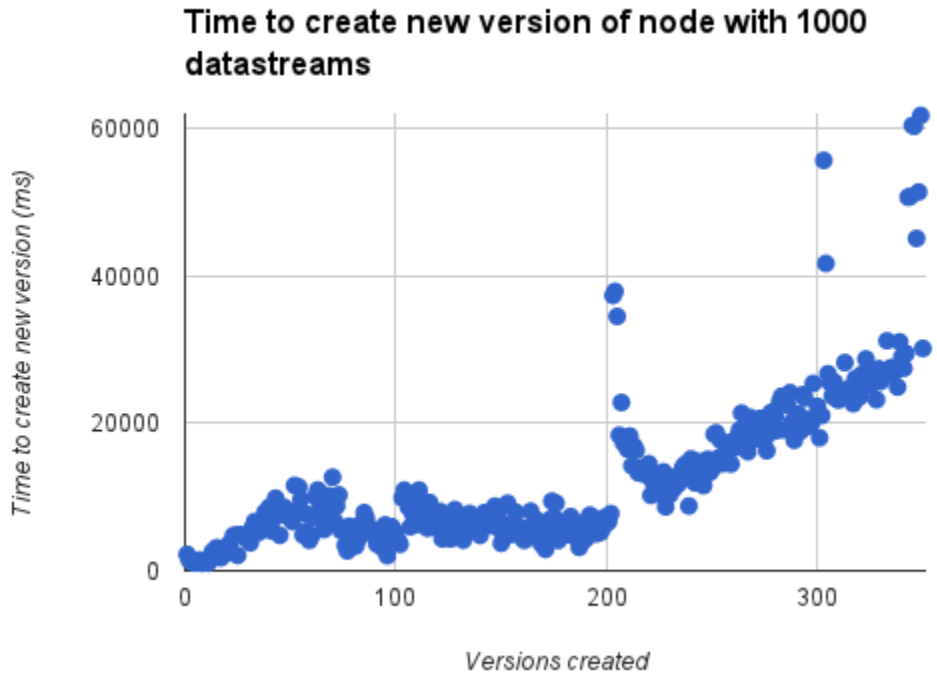
https://docs.google.com/spreadsheets/d/1SnFE-mUMEJnFUr3hXvg8UVBhnl4pq05lDTGlGR_VwYw

Multiple versions of the same container

Created a single resource with one binary and then created new versions of the resource:



Performance drop-off was considerably faster when numerous binaries (1000) were added to the root resource prior to creating many new versions of the resource.



Data

https://docs.google.com/spreadsheets/d/1QRieqQTq4LtR5r5AU0LpUPO7_C_ieBNGKU1ezqSrG5M

https://docs.google.com/spreadsheets/d/10B0ZaIrNeb0GYqQsEZszVZinXAVD1qB7xz30xuI_yws