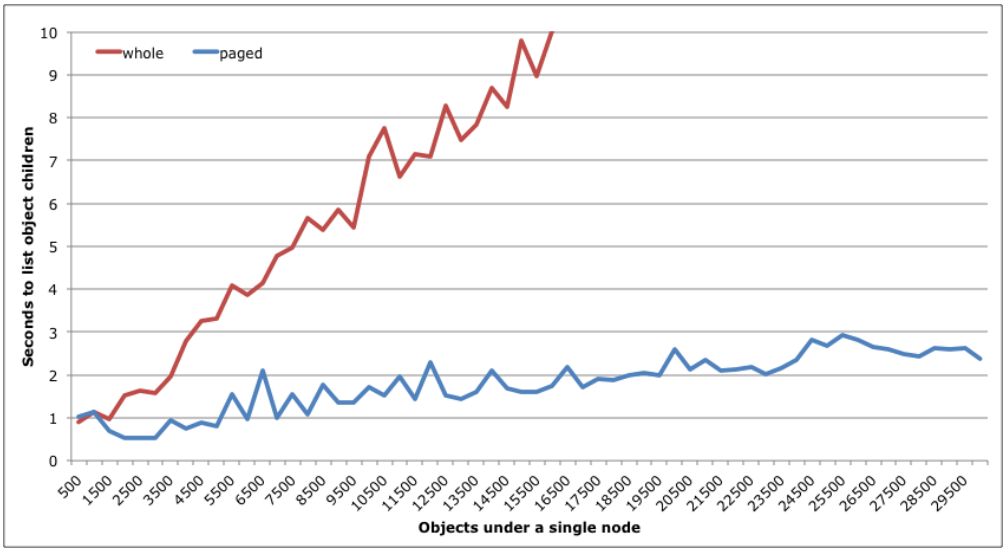
Large Numbers of Containers

Fedora 4 can handle at least 10 million containers, but performance degrades if there are too many children and/or grandchildren of a single resource.

Repository storage

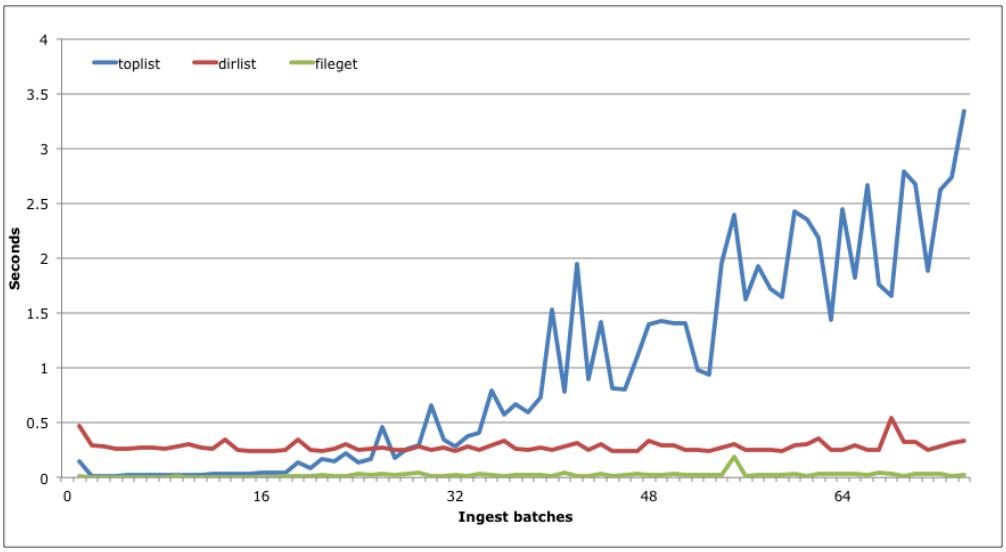
Containers under a single resource

Under a single resource, batches of 500 containers were created, each with a single 10KB binary. After each batch, the entire directory contents were listed using the REST API ("whole"). The test was then repeated using the limit parameter to list the first 1,000 containers ("paged"). Listing the containers scales roughly linearly in both cases, but much more steeply for listing all the children. Using paging, acceptance is acceptable up to 30,000 children with no sharp increase observed.



5 million files in a 3-level hierarchy

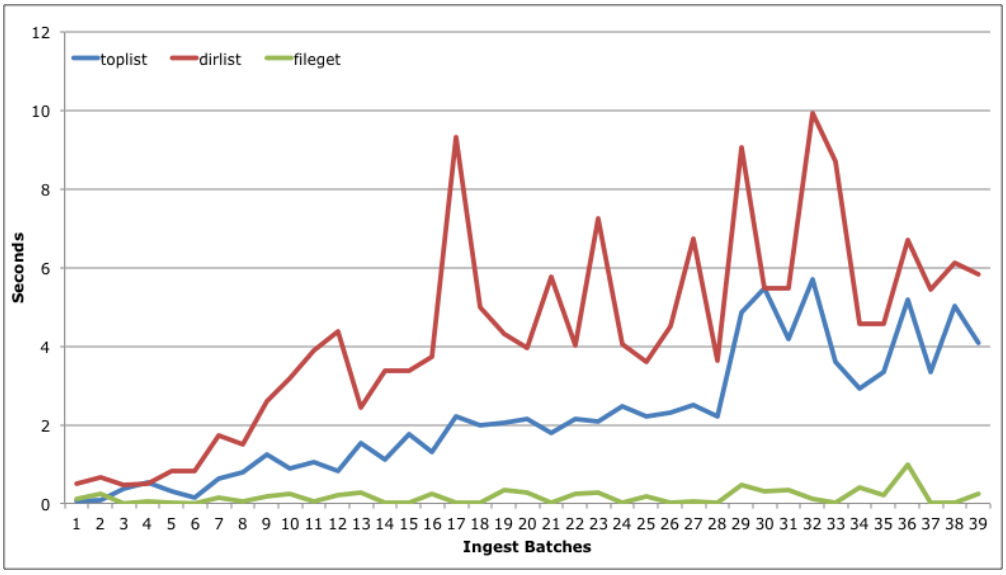
4.85 million files were ingested using a three-level hierarchy (74 top-level resources, 256 second-level resources in each, 256 third-level resources in each, and one 10KB binary in each), taking 111 hours. After each batch, three REST API operations were timed: listing the top level of the repository ("toplist"), listing a second-level resource ("dirlist"), and retrieving a file ("fileget"). Performance retrieving files and listing the second-level resources did not degrade with larger numbers of containers. However, listing the top-level of the repository degraded roughly linearly as more containers were added, and became increasing erratic.



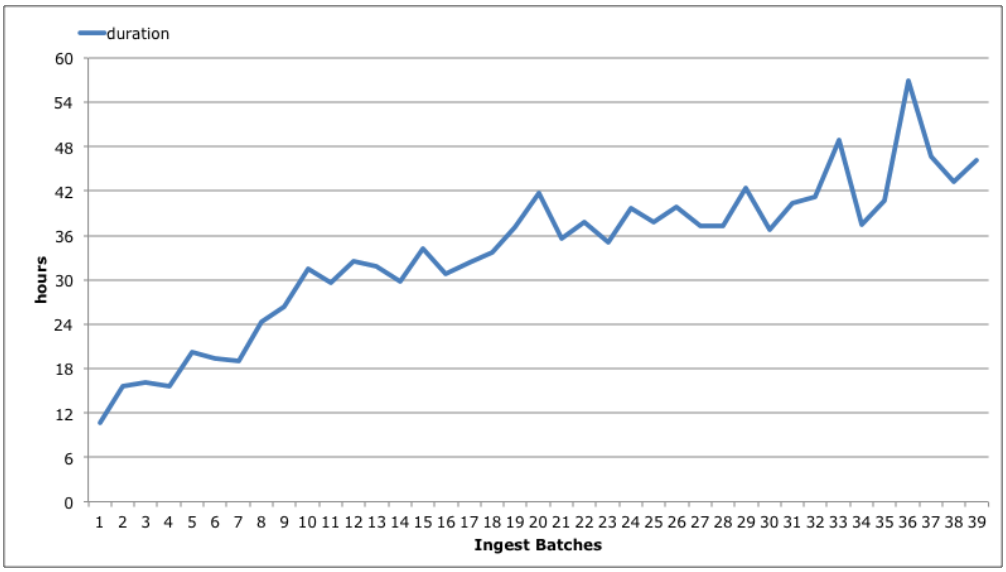
10 million files in a 4-level hierarchy

10 million files were ingested into a test repository running Fedora 4.0-beta1 (lib-devsandbox1.ucsd.edu) using a four-level hierarchy (39 top-level resources, 64 second- through fourth-level resources, and one 10KB binary in each bottom-level resource), taking 54 days (averging 186K containers /day). Ingest was done using a Java program, with a randomized binary data.

After each batch (256K containers), three REST API operations were timed: listing the top level of the repository ("toplist"), listing a third-level resource ("dirlist"), and retrieving a file ("fileget"). Performance retrieving files did not degrade with larger numbers of containers. However, listing the top-level of the repository degraded roughly linearly as more containers were added, and listing a third-level resource increased more rapidly, with increasing variability as more containers were created.



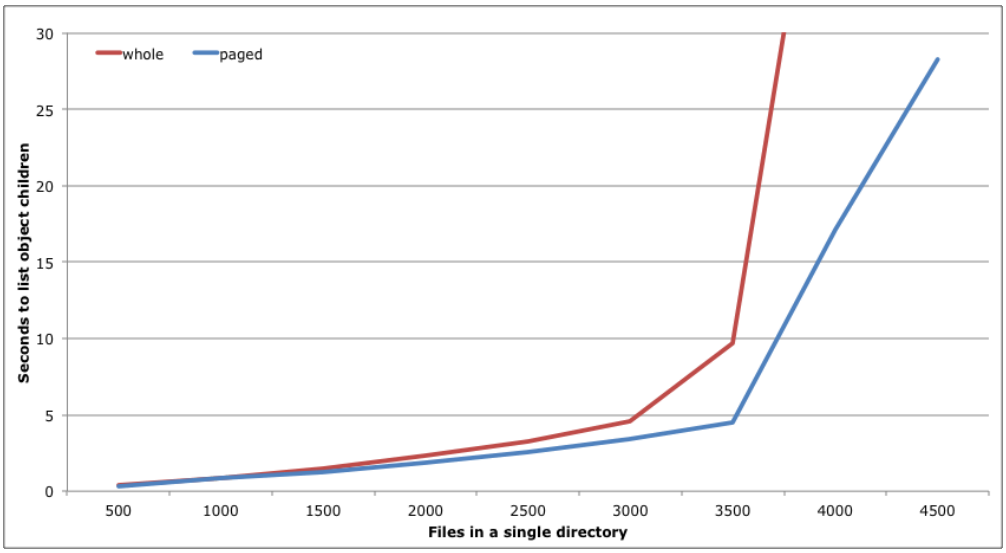
The duration of ingesting each batch of 256K containers also increased steadily:



Federated filesystem

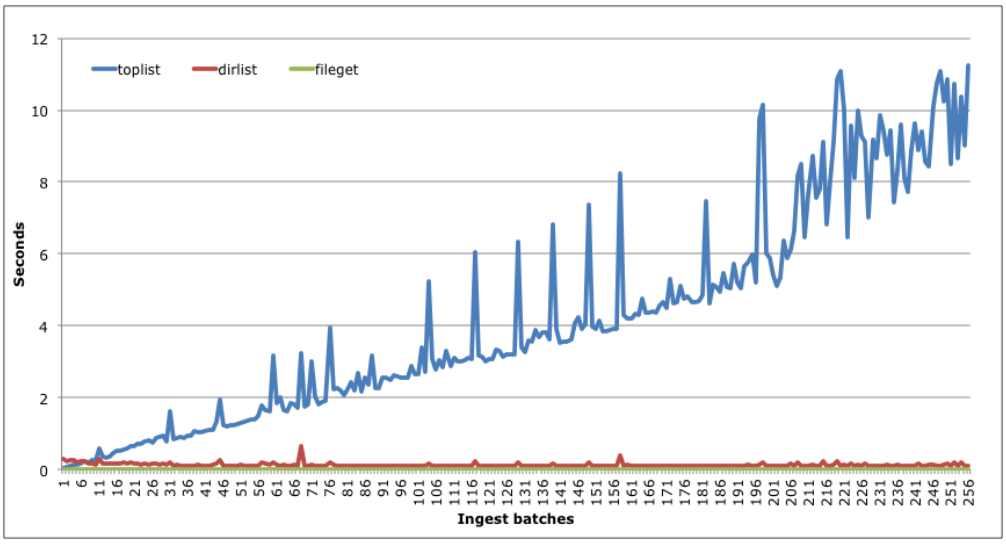
Files in a single directory

Using a single directory, batches of 500 files were created to test performance. After each batch, the entire directory contents were listed using the REST API ("whole"). The test was then repeated using the limit parameter to list the first 1,000 files ("paged"). Listing the files in the directory scales roughly linearly up to around 3,000 files, then degrades sharply. Using paging to list only 1,000 files in a batch makes the degradation a slightly less severe.



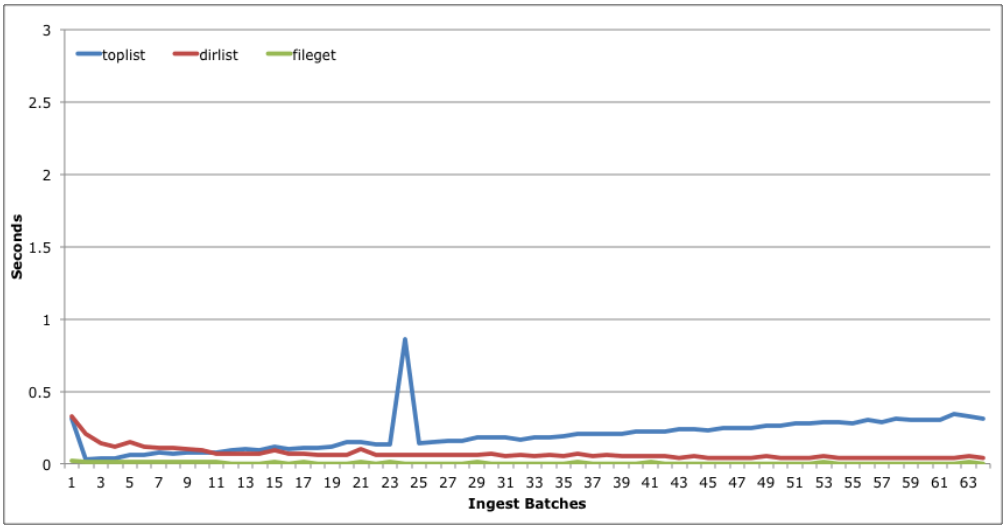
16.7 million files in a 3-level hierarchy

16.7 million files were generated on a federated filesystem using a three-level hierarchy (256 top-level resources, 256 second-level resources in each, 256 third-level resources in each, and one 10KB binary in each), taking 26.78 hours. After each batch, three REST API operations were timed: listing the top level of the repository ("toplist"), listing a second-level resource ("dirlist"), and retrieving a file ("fileget"). Performance retrieving files and listing the second-level resources did not degrade with larger numbers of containers. However, listing the top-level of the repository degraded roughly linearly as more containers were added, and became increasing erratic.



16.7 million files in a 4-level hierarchy

16.7 million files were generated on a federated filesystem using a four-level hierarchy (64 containers per level, with each leaf container having a single 10KB binary), taking 18.19 hours. Performance retrieving files and listing the third-level resources was essentially flat at less than 0.1 second. After each batch, three REST API operations were timed: listing the top level of the repository ("toplist"), listing a third-level resource ("dirlist"), and retrieving a file ("fileget"). Time to list the top-level of the repository increased linearly, but was still extremely fast even with a fully-populated hierarchy of 16.7 million containers/files – even the spike at the 24th batch was less than 1 second.



Conclusion

File retrieval performs very well, and seems to be unaffected by the number of files and the hierarchy used. The performance of using the REST API to list the contents of a directory on a federated filesystem depends on the number of children and grandchildren resources. So a 3-level hierarchy with 100-128 resources at each level should allow a federated filesystem with 1-2 million files to perform well, but larger sets of files should use a 4-level (or deeper) hierarchy.