Modeshape Unordered Collections

Modeshape 4.4.0 includes several new mixins that change the behavior for storing children of Nodes. Adding these mixins triggers storing child nodes in a progressively larger number of buckets, which is intended to improve performance with large numbers of children. See Modeshape's Unordered Large Collection documentation for more information.

Containers Tested

Name	Container	Child Nodes
PairTree	default	default auto-generated IDs
Flat	default	all children created directly in the container
Tiny	mode:unorderedTinyCollection	all children created directly in the container
Small	mode:unorderedSmallCollection	all children created directly in the container
Large	mode:unorderedLargeCollection	all children created directly in the container
Huge	mode:unorderedHugeCollection	all children created directly in the container

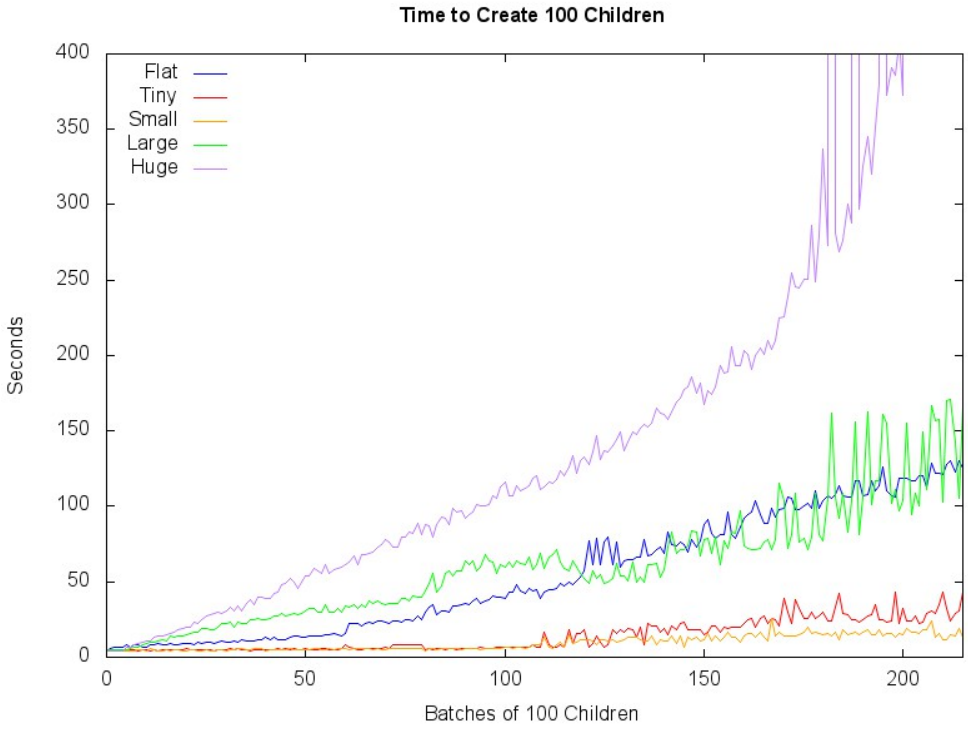
Tests

Creating Containers

For each type tested (Flat, Tiny, Small, Large, Huge), use Curl to create 100 children. Count the number of seconds to create the 100 children (write time).

- Status: Errors after 20K children created in each container
- Write Performance: Tiny and Small performed best, scaling smoothly through 20K children; Flat and Large slowed more; and Huge slowed dramatically.

Chart showing seconds to create 100 children:

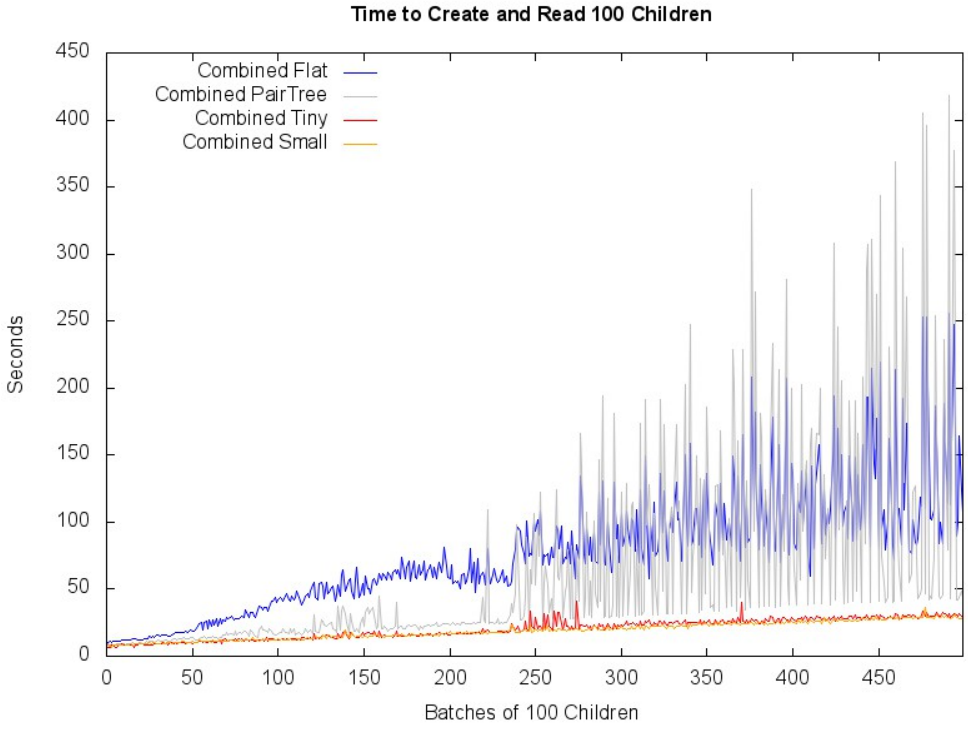


Creating and Reading Containers

For each type tested (PairTree, Flat, Tiny and Small), use Curl to create 100 children, list children in the container, and retrieve 100 children. Count the number of seconds to create the 100 children (write time), and the number of seconds to list and retrieve the children (read time).

- Status: 50K children created in each container
- Read Performance: Tiny and Small are scaling smoothly; Flat and PairTree are slower and increasingly variable after 25K children.
- Write Performance: PairTree, Tiny, and Small are scaling smoothly; Flat is slowing down much more dramatically and increasingly variable starting around 12K children.

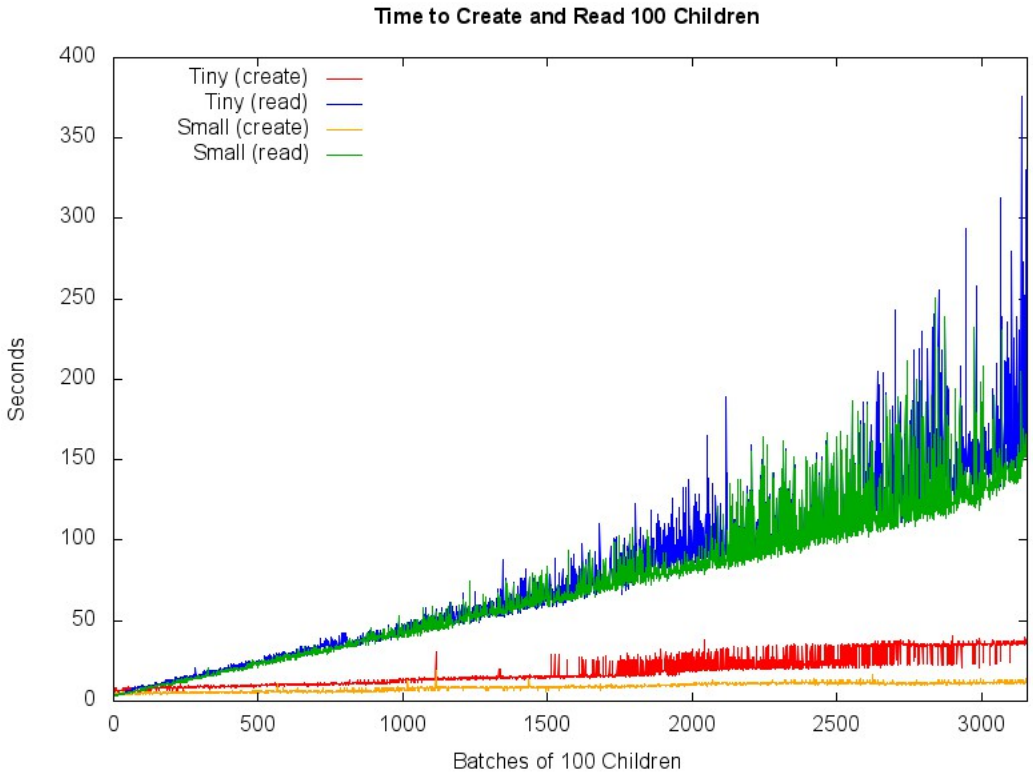
Chart showing seconds to create and read 100 children:



Creating and Reading Containers (Part 2)

Based on the previous test, the Tiny and Small unordered collections seemed the most promising. Repeat the previous test with only Tiny and Small containers, and continue testing with larger numbers of children to see how many children they can contain before performance degrades.

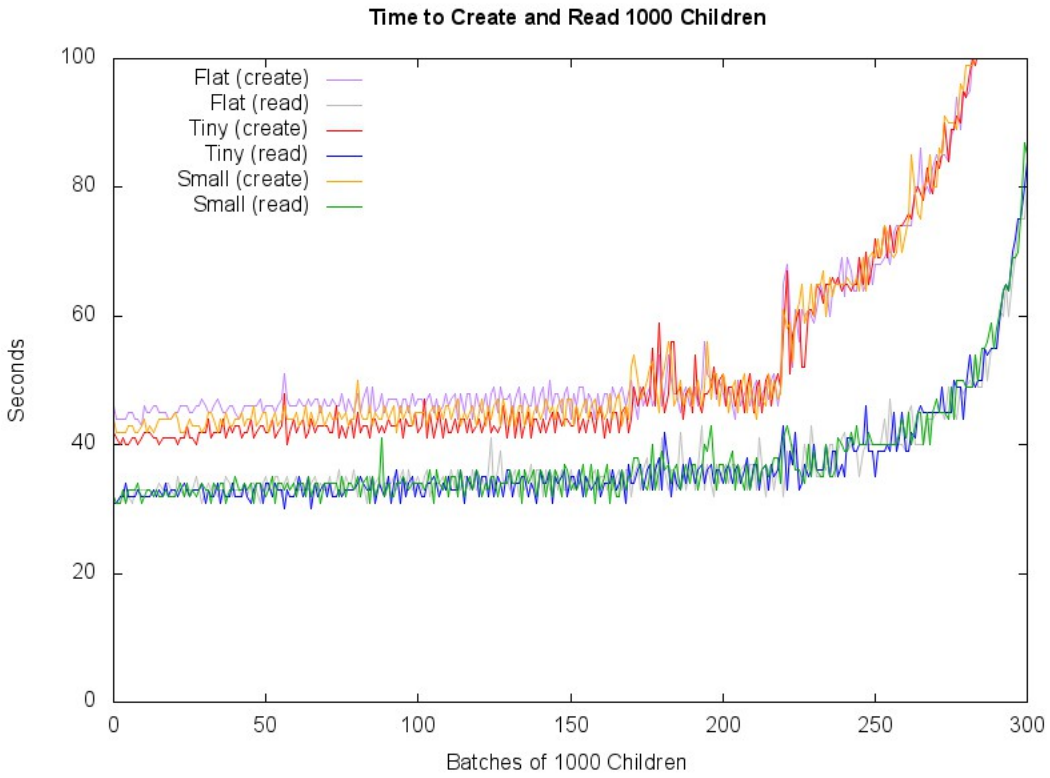
- Status: 300K children created in each container
- Read Performance: Both scaling smoothly through 100K children, and becoming more erratic afterwards, with Small performing marginally better.
- Write Performance: Both scaling smoothly, with Small performing significantly better.



Creating and Reading Containers (1-Level Hierarchy)

To see if the limitation was the number of children directly under a single container, run a new set of tests with a 1-level hierarchy, with 1000 contianers each containing 1000 children.

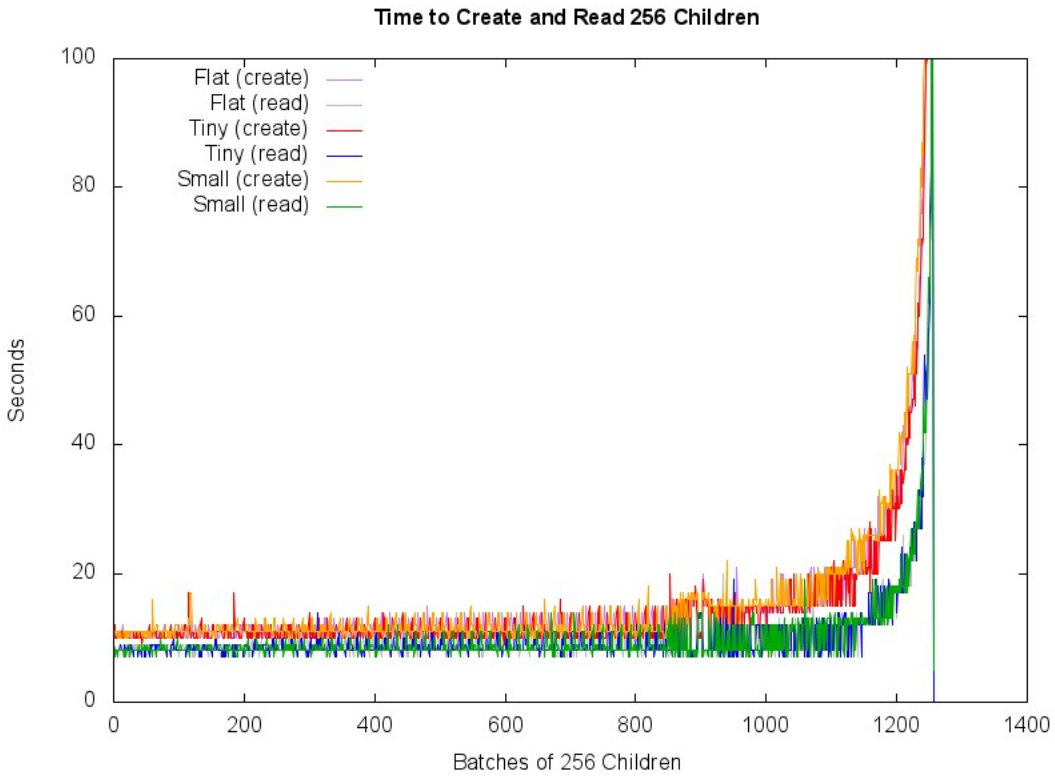
- Status: 300K children created in each container
- Read Performance: Very little difference between the different container types, with performance degrading sharply after about 275K children.
- Write Performance: Very little difference between the different container types, with performance degrading sharply after about 225K children.



Creating and Reading Containers (2-Level Hierarchy)

To see if a deeper hierarchy would improve performance, run a new set of tests with a 2-level hierarchy, with 256 containers, each containing 256 child containers, each containing 256 children.

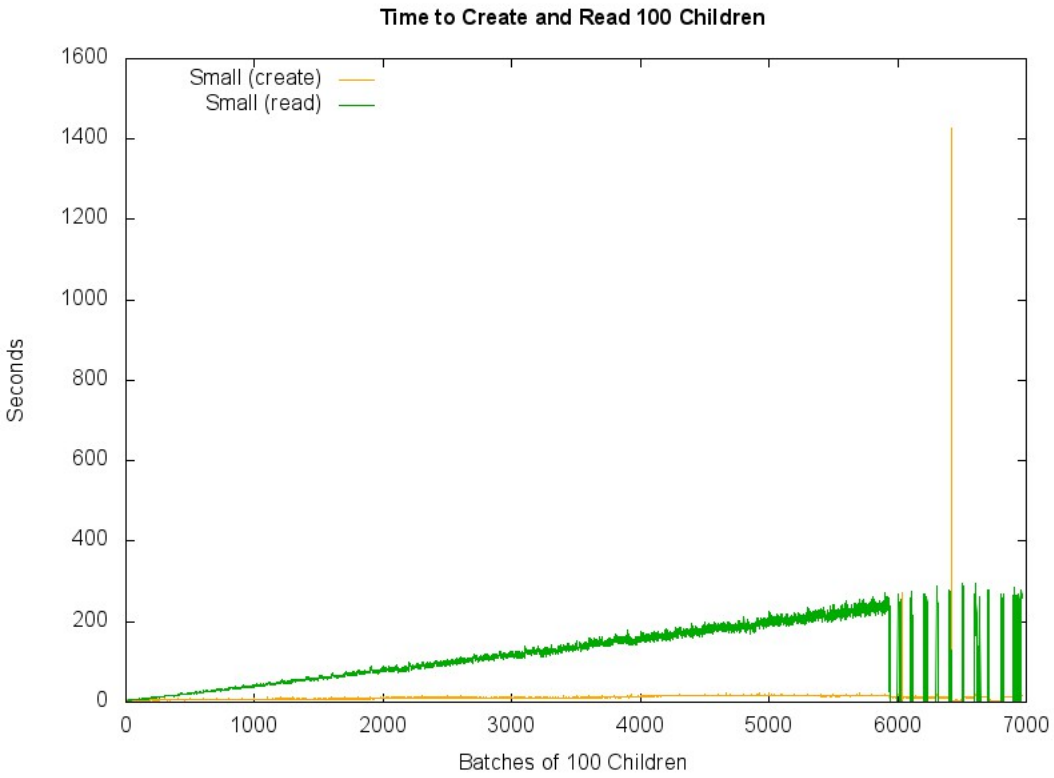
- Status: 300K children created in each container
- Read Performance: Very little difference between the different container types, with performance degrading sharply after about 1150 batches (295K children).
- Write Performance: Very little difference between the different container types, with performance degrading sharply after about 1150 batches (295K children).



Creating and Reading Containers (3-Level Hierarchy, 100 Nodes per Level)

To see if using multiple container types in the same tests and repository was impacting other container types, run a new set of tests with only Small containers, with a 2-level hierarchy, with 100 top-level containers, each containing 100 child containers, each containing 100 children.

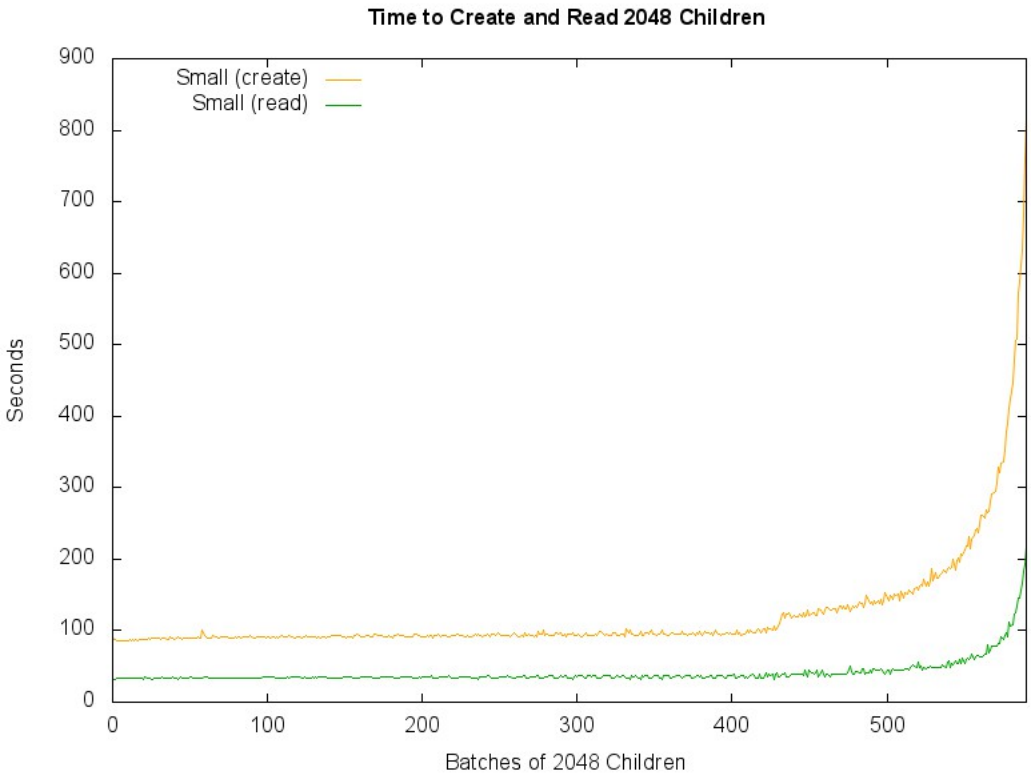
- Status: 600K children created
- Read Performance: Steadily increased until around 600K children, then repository failure.
- Write Performance: Roughly constant with increasing repository size.



Creating and Reading Containers (2-Level Hierarchy, 2K Nodes per Level)

Trying a flatter hierarchy, with a larger number of child nodes at each level: 2048 top-level containers, each with 2048 children.

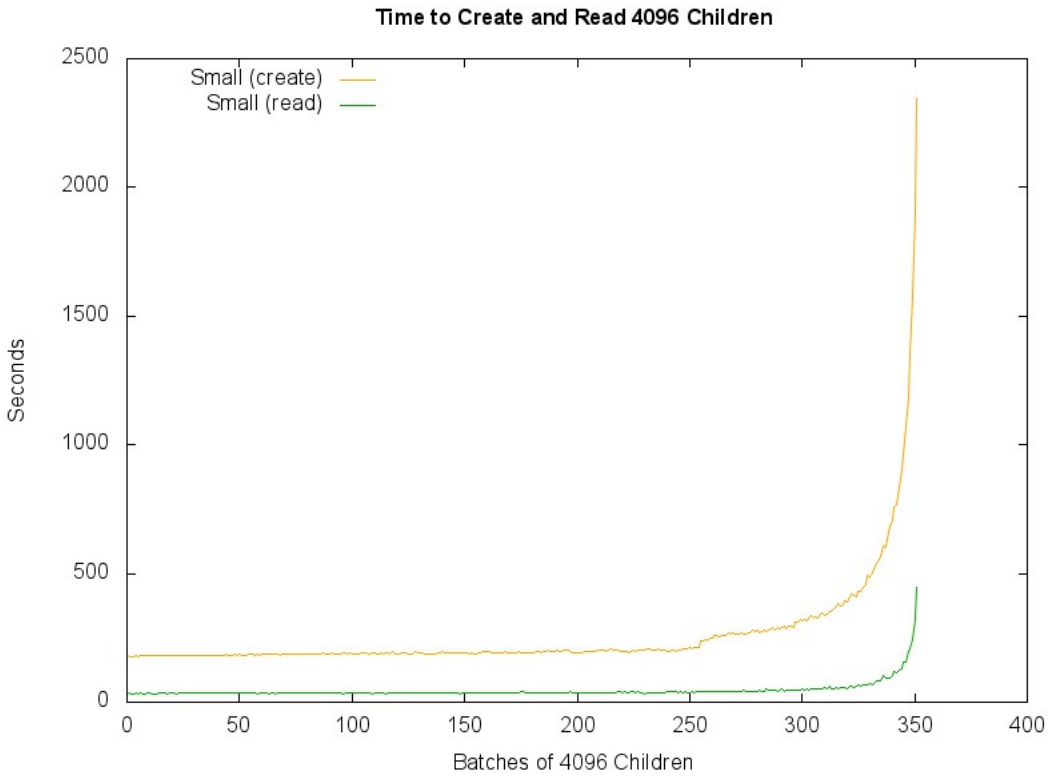
- Status: 1.2M children created
- Read Performance: Very slowly increasing until around 1.2M children, then dramatically increasing.
- Write Performance: Very slowly increasing until around 1.2M children, then dramatically increasing.



Creating and Reading Containers (2-Level Hierarchy, 4K Nodes per Level)

Trying a larger number of child nodes at each level, 4096 top-level containers, each with 4096 children.

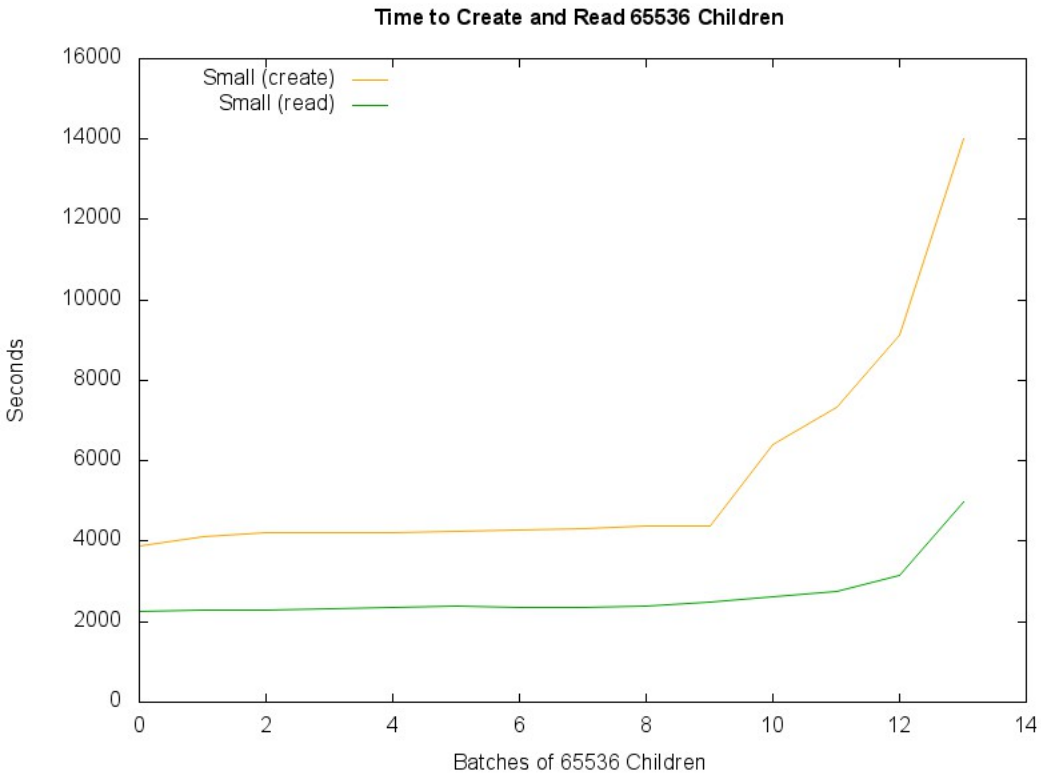
- Status: 1.4M children created
- Read Performance: Very slowly increasing until around 1.4M children, then dramatically increasing.
- Write Performance: Very slowly increasing until around 1.4M children, then dramatically increasing.



Creating and Reading Containers (2-Level Hierarchy, 64K Nodes per Level)

Trying a larger number of child nodes at each level, 65,536 top-level containers, each with 65,536 children.

- Status: 850K children created
- Read Performance: Very slowly increasing until around 590K children, then sharply increasing.
- Write Performance: Very slowly increasing until around 785K children, then sharply increasing.


Time to Create and Read 65536 Children
Small (create)
Small (read)
Seconds
Batches of 65536 Children
0
2000
4000
6000
8000
10000
12000
14000
16000
0
2
4
6
8
10
12
14