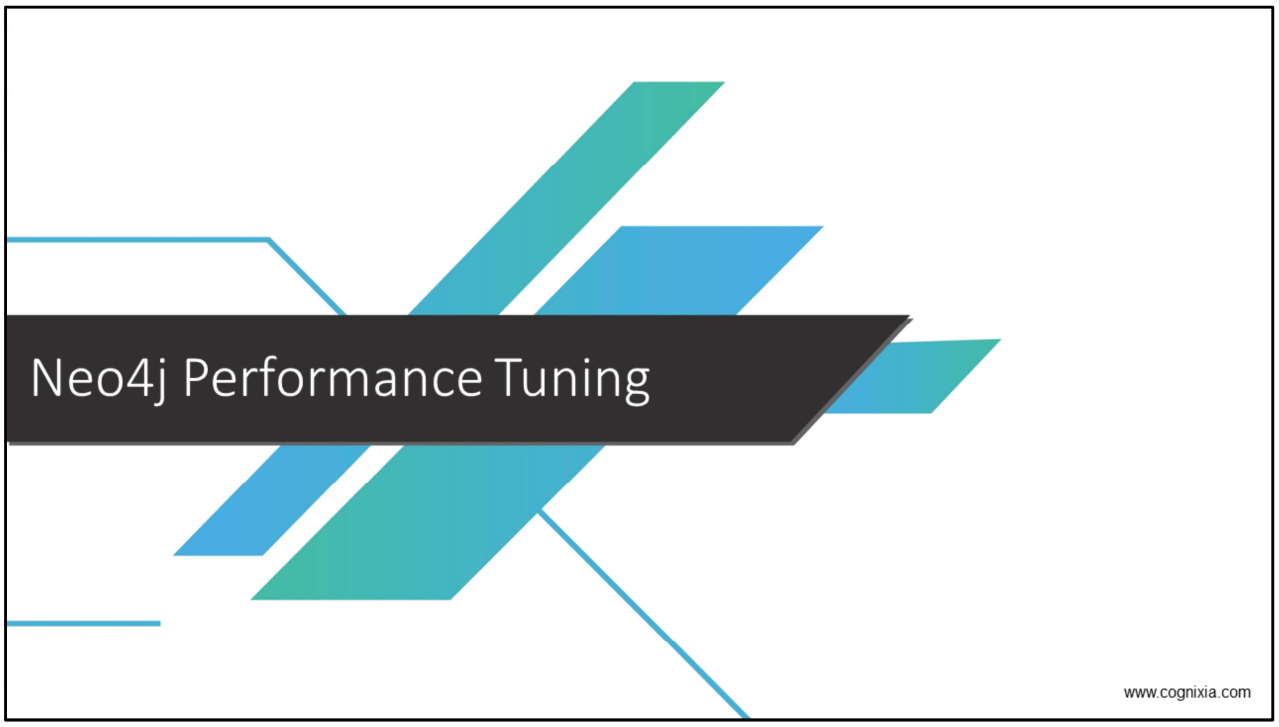
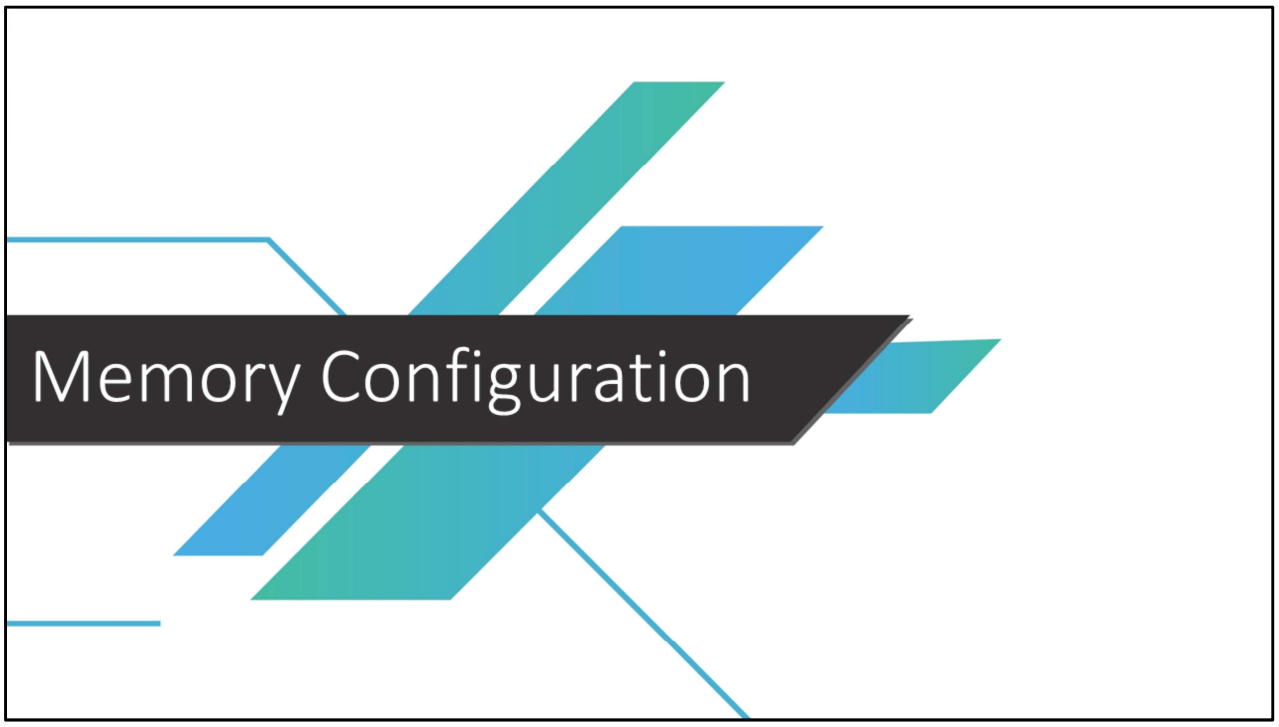
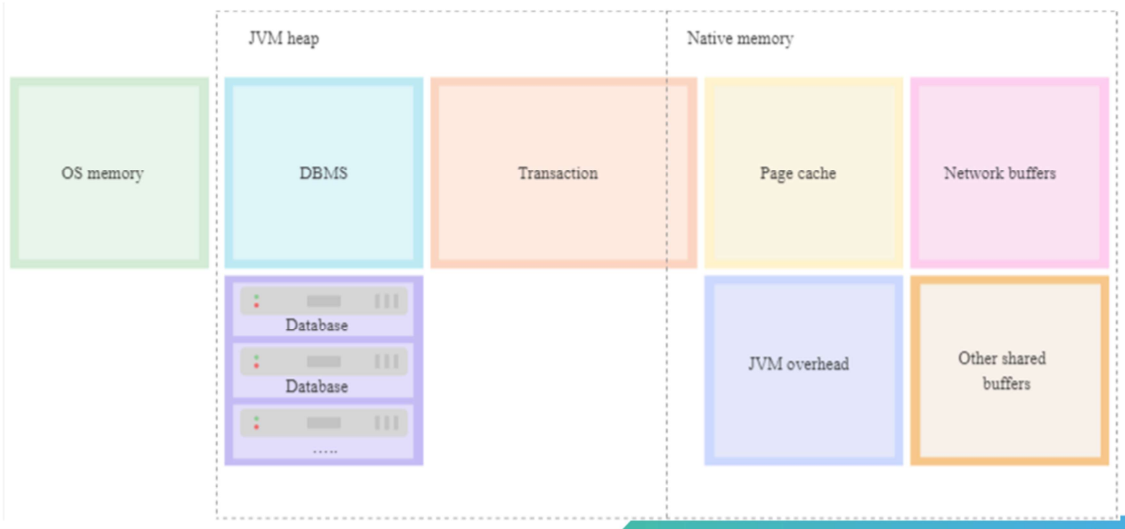

Neo4j Performance Tuning
www.cognixia.com


Memory Configuration

Memory configuration

- RAM of the Neo4j server has a number of usage areas



www.cognixia.com

Considerations

- Always use explicit configuration
  - Recommended to always define the page cache and heap size parameters explicitly
- Initial memory recommendation
  - Use **neo4j-admin memrec** command to get an initial recommendation
- Inspect the memory settings of all databases
  - **neo4j-admin memrec** command is useful for inspecting

shell

```
$neo4j-home> bin/neo4j-admin memrec
...
...
...
# Total size of lucene indexes in all databases: 6690m
# Total size of data and native indexes in all databases: 17050m
```

Capacity planning

- It is advantageous to try to cache as much of the data and indexes as possible

```
$neo4j-home> bin/neo4j-admin memrec
...
...
...
# Total size of lucene indexes in all databases: 6690m
# Total size of data and native indexes in all databases: 35050m
```

- dbms.memory.pagecache.size = 1.2 * (35GB) = 42GB

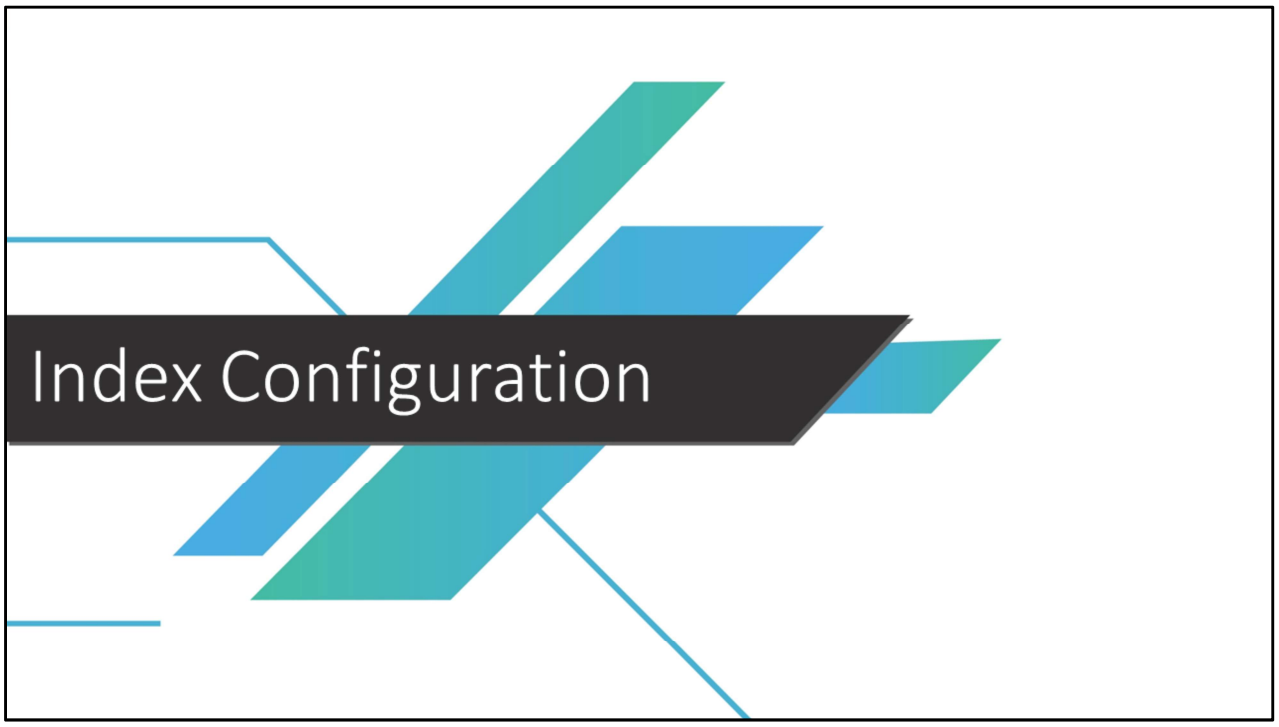
Limit transaction memory usage

- dbms.memory.transaction.global_max_size=256m
  - Global maximum memory usage for all of the transactions
  - Must be configured low enough so that you do not run out of memory
- dbms.memory.transaction.database_max_size=100m
  - Limits the transaction memory usage per database.
- dbms.memory.transaction.max_size=10m
  - constrains each transaction

- When any of the limits are reached, the transaction is terminated without affecting the overall health of the database.

Limit transaction memory usage

- To help configure these settings you can use the following commands to list the current usage:
  - CALL dbms.listPools()
  - CALL dbms.listTransactions()
  - CALL dbms.listQueries()


Index Configuration

Introduction

- Two different index types
  - b-tree and
  - full-text

Index migration

Index provider in 3.5	Index provider in 4.1.3
native-btree-1.0	native-btree-1.0
lucene+native-2.0	native-btree-1.0
lucene+native-1.0	lucene+native-3.0
lucene-1.0	lucene+native-3.0

Procedures to create index

- :play movie-graph # Create some sample data
- CALL db.indexes()
- CALL db.createIndex("MyIndex", ["Person"], ["name"], "native-btree-1.0")
- CALL db.indexes()
- DROP INDEX MyIndex
- CALL db.createUniquePropertyConstraint("MyIndex", ["Person"], ["name"], "native-btree-1.0")
- DROP INDEX MyIndex
- CALL db.createNodeKey("MyIndex", ["Person"], ["name"], "native-btree-1.0")
- CALL db.indexes()


Tuning of garbage collector

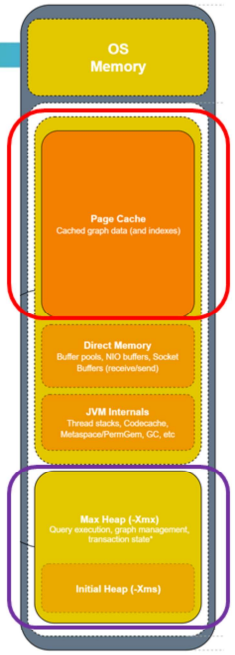
Introduction

- The effect of the Java Virtual Machine's garbage collector with regards to Neo4j performance.

- Memory leaks can happen, more often than not, a higher memory consumption is normal behaviour by the JVM

Neo4j's memory

- Heap
  - Runtime data area from which memory for all class instances and arrays are allocated
  - Heap storage for objects is reclaimed by garbage collector (GC)
- Off-Heap
  - Sometimes heap memory is not enough, especially when we need to cache a lot of data without increasing GC pauses
  - Off-heap memory is one of possible solutions
  - Off-heap store continues to be managed in memory and also not subject to GC
- Page cache
  - Used to cache the Neo4j data
  - Avoid costly disk access and result in optimal performance



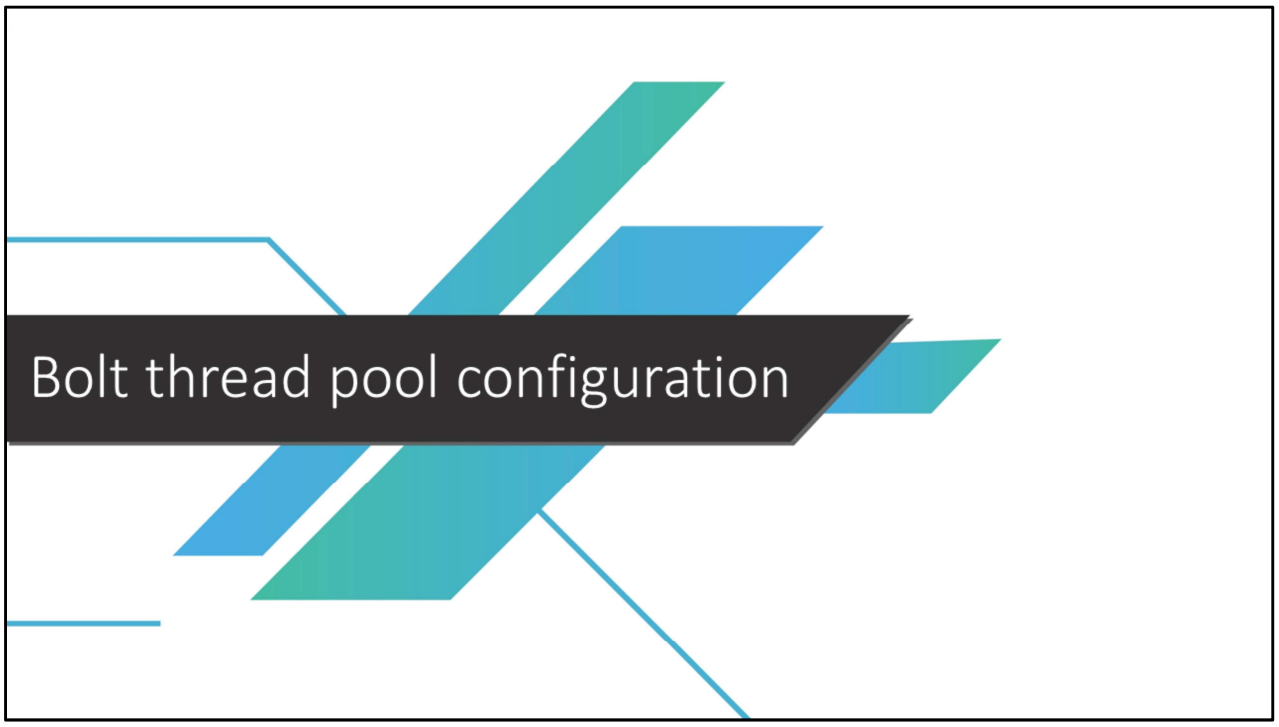
Neo4j's memory

- We can divide the Neo4j's memory consumption into 2 main areas
  - On-heap
  - Off-heap
- On-heap
  - The runtime data lives
  - Query execution, graph management and transaction state exist
- Off-heap (3 categories)
  - Neo4j's page cache
    - For caching the graph data into memory
  - Other memory the JVM needs to work (JVM Internals)
  - Direct memory

Memory settings

- vim /var/lib/neo4j/conf/neo4j.conf
- #Initial heap size
- dbms.memory.heap.initial_size=512m
- #Maximum heap size
- dbms.memory.heap.max_size=512m

- It is recommended to set the initial heap size and the maximum heap size to the same value


Bolt thread pool configuration

Introduction

- The Bolt connector is backed by a thread pool on the server side
- The thread pool is constructed as part of the server startup process.

How thread pooling works

- Has a minimum and a maximum capacity
- Idle threads are stopped and removed from the pool
- Each message arriving on a connection triggers the scheduling of a connection on an available thread in the thread pool
- If all the available threads are busy, and there is still space to grow, a new thread is created and the connection is handed over to it for processing
- If the pool capacity is filled up, and no threads are available to process, the job submission is rejected

Configuration options

Option name	Default	Description
dbms.connector.bolt.thread_pool_min_size	5	The minimum number of threads that will always be up even if they are idle.
dbms.connector.bolt.thread_pool_max_size	400	The maximum number of threads that will be created by the thread pool.
dbms.connector.bolt.thread_pool_keep_alive	5m	The duration that the thread pool will wait before killing an idle thread from the pool. However, the number of threads will never go below dbms.connector.bolt.thread_pool_min_size.

www.cognixia.com

dbms.connector.bolt.thread_pool_min_size=5
dbms.connector.bolt.thread_pool_max_size=400
dbms.connector.bolt.thread_pool_keep_alive=5m


Linux file system tuning

Linux file system tuning

- Databases often produce
  - Many small and random reads when querying data, and
  - Few sequential writes when committing changes
- Recommended to store database and transaction logs on separate devices
- The EXT4 and XFS file systems are recommended as a first choice.

- Recommended practice is to disable **file and directory access** time updates
- This way, the file system won't have to issue writes that update this meta-data, thus improving write performance

Storage and RAM

- High read and write I/O load degraded SSD performance.
- Ensure that the working dataset fits in RAM
- To be able to achieve optimum performance, it's not recommend the use of NFS or NAS as database storage.

Disks, RAM & other tips

- To achieve maximum performance, it is recommended to provide Neo4j with as much RAM as possible in order to avoid hitting the disk.

- **vmstat** - Gather information when application is running
- High swap usage is a sign that the database don't fit in memory
- In this case, database access can have high latencies


Statistics & execution plans

Introduction

- When a Cypher query is issued, it gets compiled to an execution plan
- Neo4J and Cypher query engine uses available information about the database

Statistics

- The statistical information that Neo4j keeps is:
  - # of nodes having a certain label.
  - # of relationships by type.
  - Index details

Statistics

- Control whether statistics are collected automatically

Parameter name	Default value	Description
dbms.index_sampling.background_enabled	true	Controls whether indexes will automatically be re-sampled The Cypher query planner depends on accurate statistics to create efficient plans, so it is important it is kept up to date
dbms.index_sampling.update_percentage	5	Controls the percentage of the index that has to have been updated before a new sampling run is triggered.

dbms.index_sampling.background_enabled=true
dbms.index_sampling.update_percentage=5
neo4j restart && neo4j status && tail -f -n 50 /logs/debug.log && curl localhost:7474

Statistics

- It is possible to manually trigger index sampling, using the following built-in procedures:
  - # Trigger resampling of an index
  - CALL db.resampleIndex("MyIndex");
  - # Trigger resampling of all outdated indexes
  - CALL db.resampleOutdatedIndexes();

CALL db.resampleIndex("MyIndex");
CALL db.resampleOutdatedIndexes();

Execution plans

- Execution plans are cached by default

Parameter name	Default value	Description
cypher.statistics_divergence_threshold	0.75	• The threshold when a plan is considered outdated • If any of the underlying statistics used to create the plan have changed more than this value, the plan will be considered outdated and will be replanned • A value of 0 means replan as soon as possible
cypher.min_replan_interval	10s	• The minimum time between cypher query replanning events • After this time, the graph statistics will be evaluated, and if they have changed by more than the value set by cypher.statistics_divergence_threshold, the query will be replanned

www.cognixia.com

vim /var/lib/neo4j/conf/neo4j.conf
cypher.statistics_divergence_threshold=0.75
cypher.min_replan_interval=10s
neo4j restart && neo4j status && tail -f -n 50 /logs/debug.log
curl localhost:7474

Manually force

- # Clears out all query caches, but does not change the database statistics.
  - CALL db.clearQueryCaches();
- Completely recalculates all database statistics
  - CALL db.prepareForReplanning();


Cognixia®
THANK YOU
45%