
Making
Cheetah
Faster

What is **Cheetah?**

- In-Memory Database (Java)
- Developed by *Alan Lu*
- JSON ~> Table
- Exploration of various *data stores*
- Queried via **SQL** statements
- *Single-threaded...*

Why Cheetah?

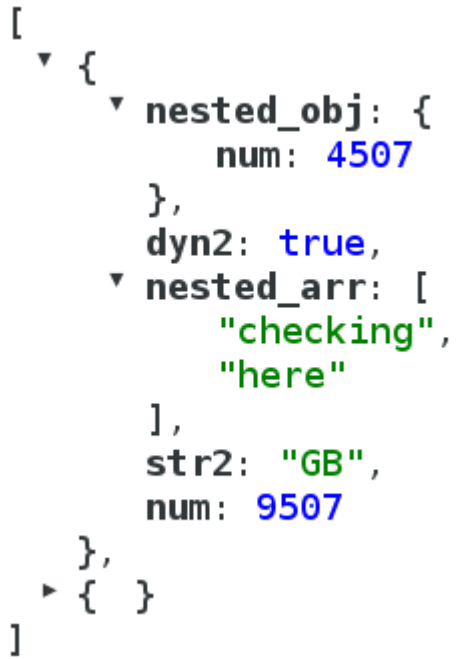
- Faster than *disk-optimized* databases (~6x)
- Eliminates *seek time* when querying...
- *Faster* and *more predictable* performance
- Critical for *real-time* response

Overview: Cheetah

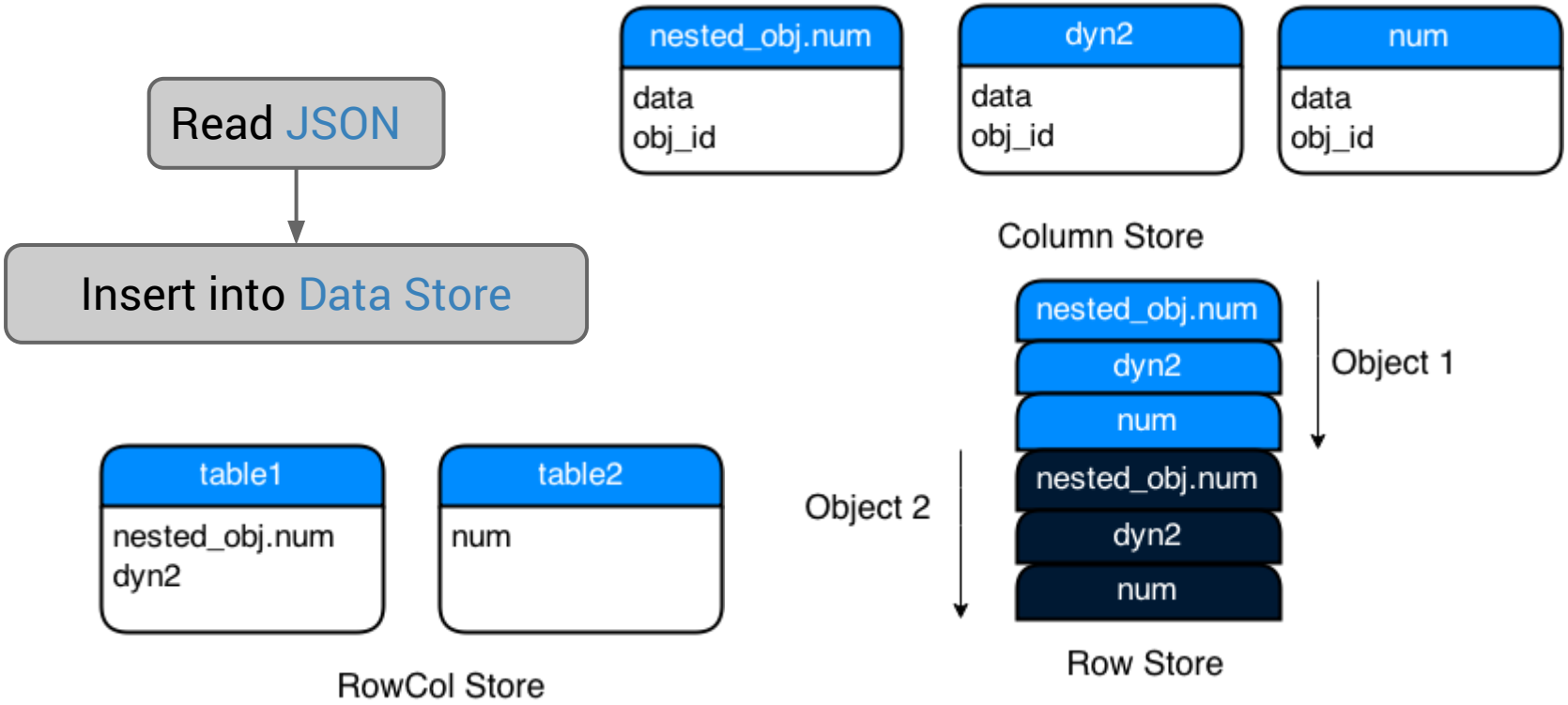
- Current *Architecture*
- Current *Data Stores*
- Exploring the Column Store Implementation
- Speeding Up *Queries* and *Insertion*!

Cheetah: Current Architecture

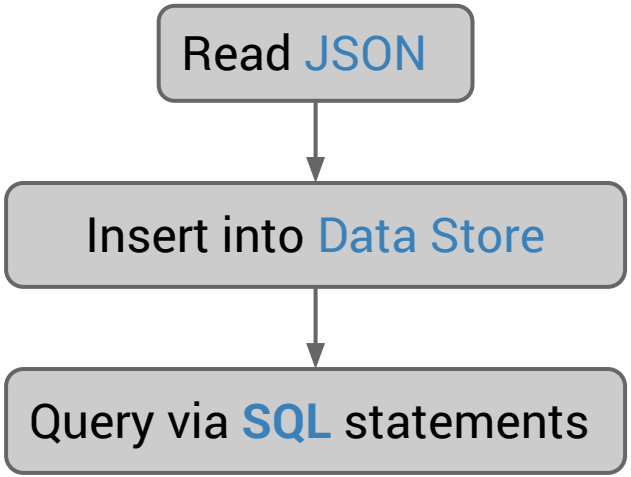
Read JSON



Cheetah: Current Architecture

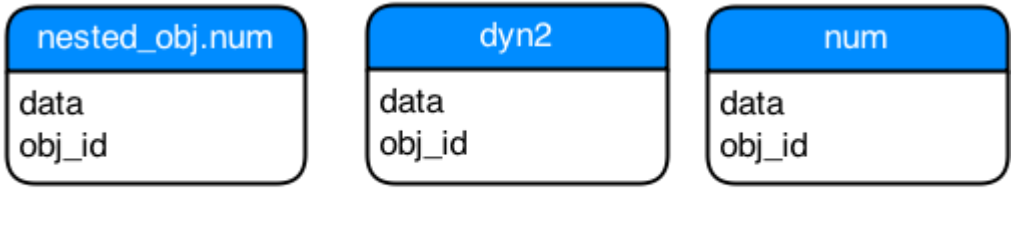


Cheetah: Current Architecture



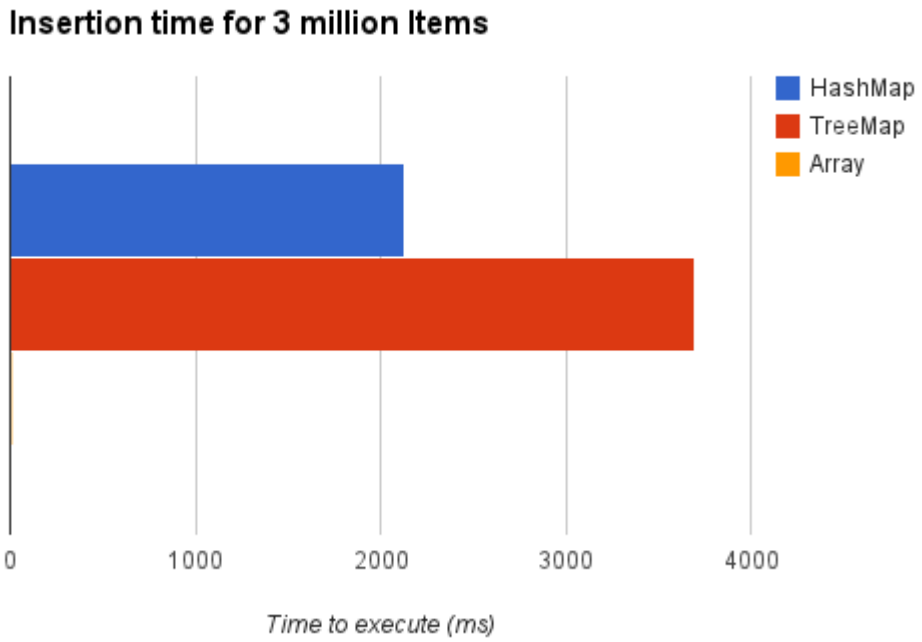

```
SELECT dyn2
WHERE  num
BETWEEN 0 AND 10
```

Cheetah: Current Data stores

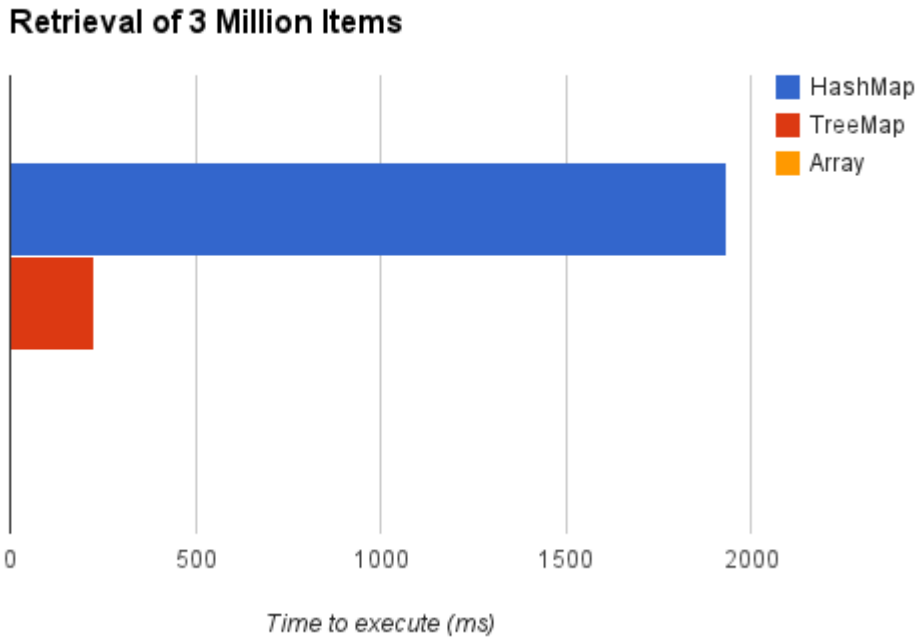


- Each *key* has a table
- No Schema required
- Uses two arrays per table
- *Excellent* for sparse data
- Not so great for *Querying*

Cheetah: Other Data Structures?



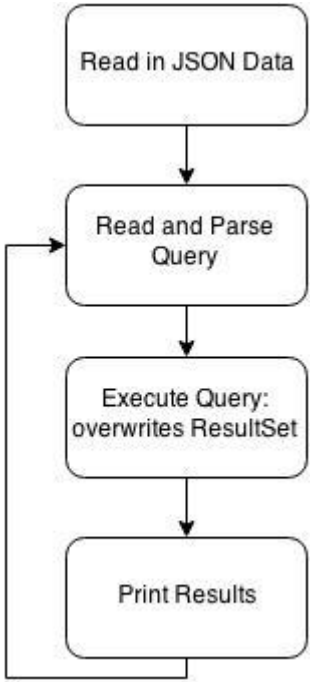
Cheetah: Other Data Structures?



Cheetah: Running Queries in Parallel

Original program flow:

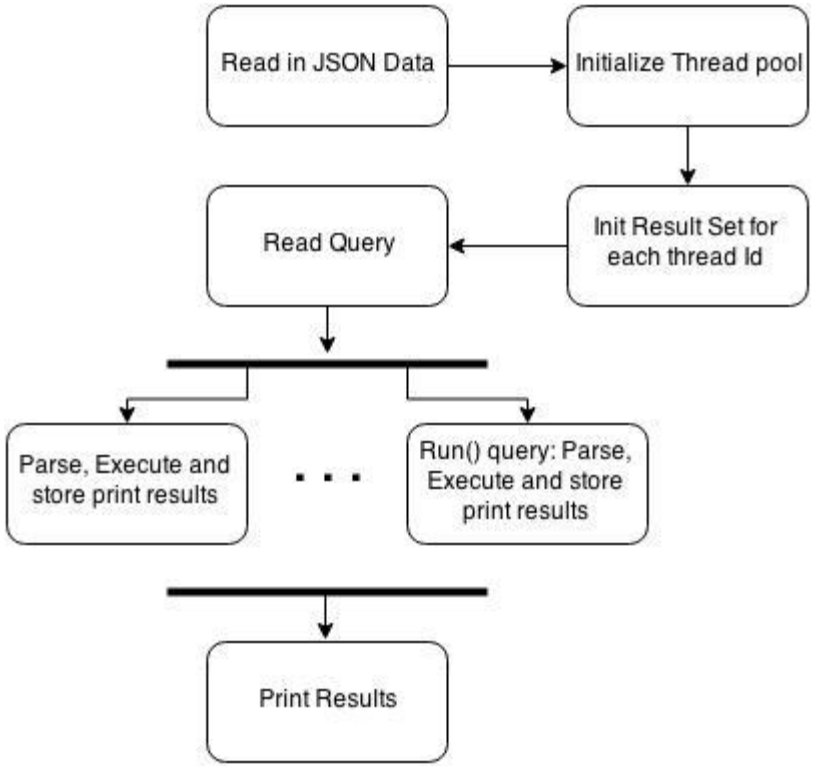
1) Read in database
2) Read and perform queries one by one
   a) Parse query
   b) Execute query → writes to a result set
   c) Print result information
3) Next query clears and overwrites result set



Cheetah: Running Queries in Parallel

Parallel program flow:

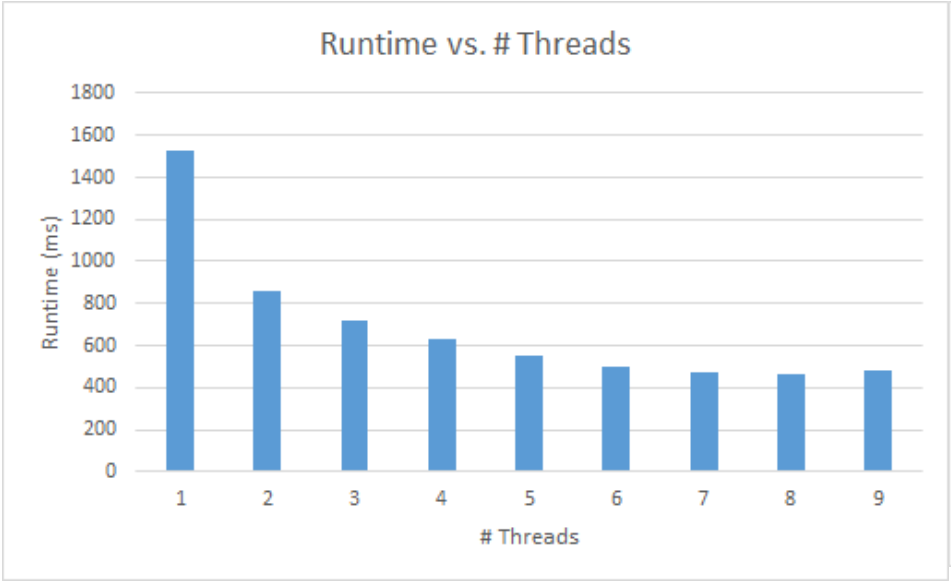
1) Read in database
2) Initialize thread pool
3) Initialize result set: each thread has it's own result set
4) Run queries in parallel
5) Print results

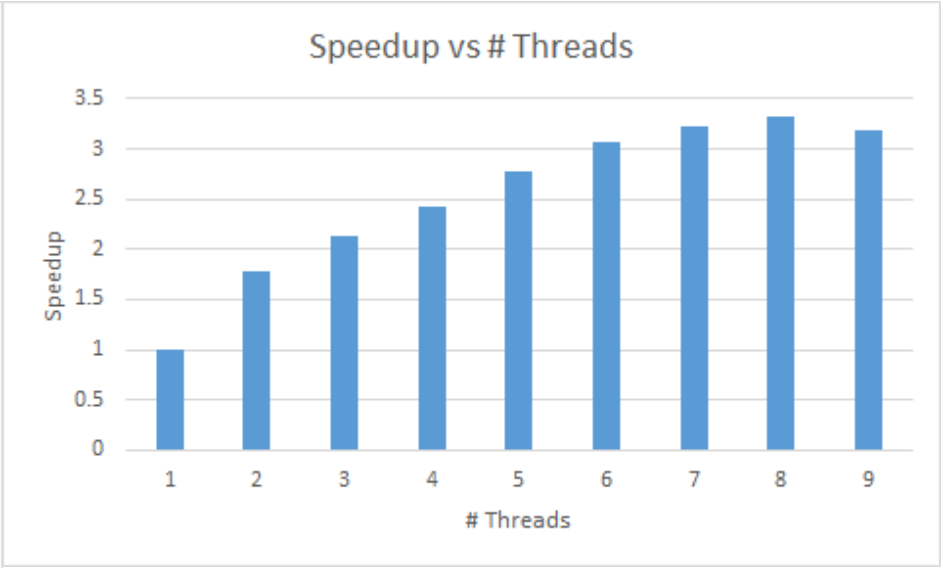


Cheetah: Issues with parallelization

- Results Sets are large: 3 large arrays in each result set
- Lack of memory

Cheetah: Running Queries in Parallel





Cheetah: Java Profiling with YourKit

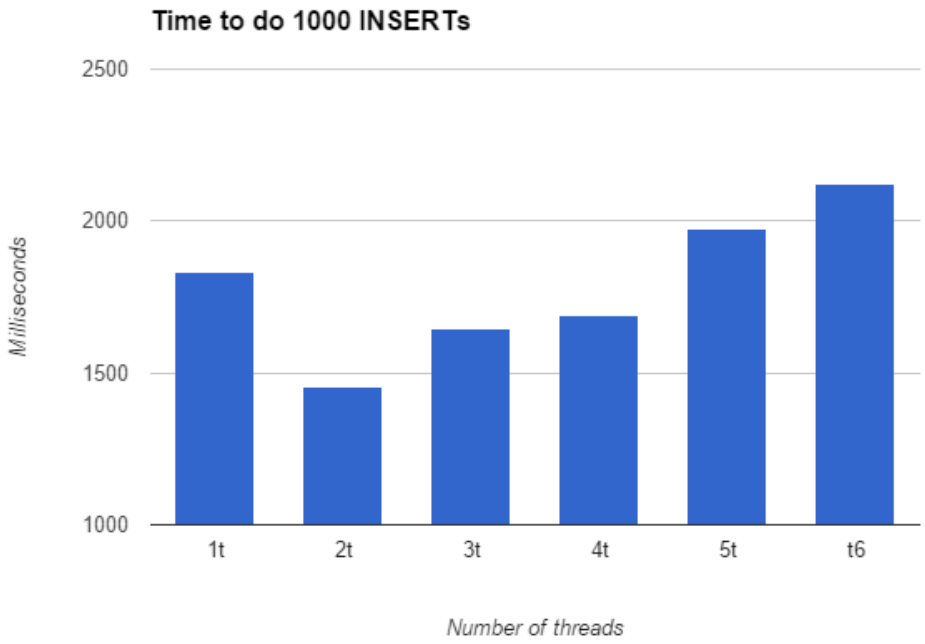
Sequential Execution time: ~7000ms

Threadpool with 1 thread time: ~1500ms

	% of time for Original program (Sequential)	% of time for parallelized program (with 1 thread)
Executing Querying	9%	~0%
Printing Summary	47%	15%

Cheetah: Faster Insertions

- Parallel inserts of JSON objects
- Coarse grained lock
- Fine grained lock works better for sparse data



Cheetah: What's next?

- Exclusion SQL statement (e.g `NOT IN`)
- Explore other data store configurations
- Cache hit/miss mechanism
- Explore lock granularities: table, column, hybrid

Cheetah: Conclusion

- Explored different data structures
- Added functionality for **SQL** `INSERT` statement
- Achieved speedup by adding multi-threaded support for read and write


Thanks for listening!