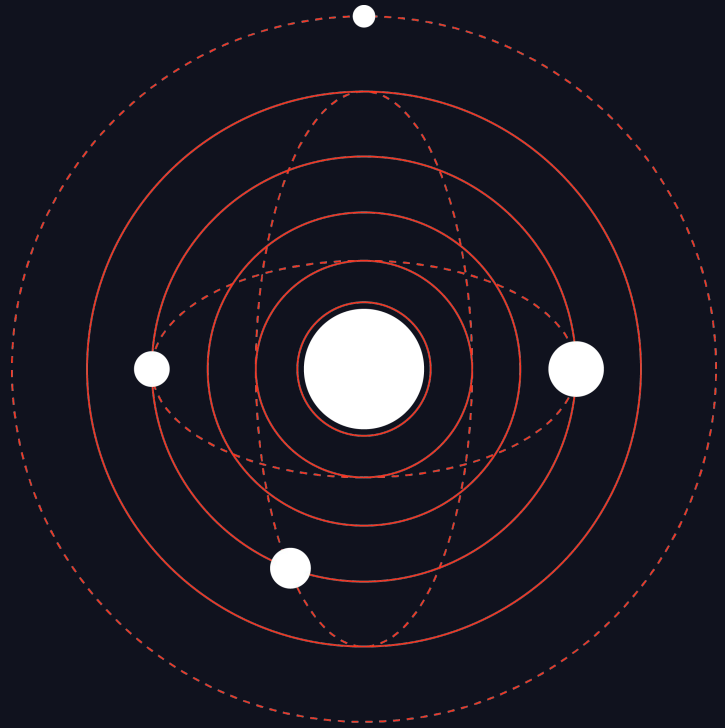
TECH SUMMIT



THREAD ZEPPELIN

An intelligent multi-threaded function-level code profiler

Project Owner: Robert Altmiller, RSA

Presenter: Kayla Grieme, Sr. Solutions Architect

Code Contributors: Alan Reese, RSA and Divyank Jain, RSA

TODAY'S PROFILER PROBLEMS

Analyzing large code bases for optimization opportunities can be a tedious and time-consuming process

Line-by-line runtime output

➡ Difficult to interpret; Slow execution

50-60% overhead to run

➡ High costs; Inefficient

Output is **not prescriptive** with too much detail

➡ Not actionable; Slow time to value

TODAY'S PROFILER PROBLEMS

Analyzing large code bases for optimization opportunities can be a tedious and time-consuming process

Line-by-line runtime output → Difficult to interpret; Slow execution

50-60% overhead to run → High costs; Inefficient

Output is **not prescriptive** with too much detail → Not actionable; Slow time to value

TODAY'S PROFILER PROBLEMS

Analyzing large code bases for optimization opportunities can be a tedious and time-consuming process

Line-by-line runtime output ➡ Difficult to interpret; Slow execution

50-60% overhead to run ➡ High costs; Inefficient

Output is **not prescriptive** with too much detail ➡ Not actionable; Slow time to value

Introducing... *Thread Zeppelin*

Embark on a journey with *Thread Zeppelin*, where we leverage the power of Spark to unravel the "Whole Lotta Love" hidden in your codebase.

- ➔ **Function level metrics** for faster identification of problem code
- ➔ Leverages easy-to-use profiling **decorator**
- ➔ Reduced overhead with **function level profiling**
- ➔ Flexible and efficient output stored in **Delta**, with a **compressed version** of the original source code
- ➔ Prescriptive AI-assisted **optimization guidance** and **code** provided using **DBRX**

Thread Zeppelin – Current State

Databricks on Databricks

- Profiler accessible **today** via Github for Bricksters & customers
- Battle-tested on a large ERP customer's code base which identified optimizations that led to an **83% reduction in workflow runtime.**
  - Runtime **Before** Optimizations = **3.3 hours** to complete for 5 customers.
  - Runtime **After** Optimizations = **1/2 hour** to complete for 5 customers.
- Projected benefit for 3000 onboarded customers:
  - Total Workflow **Daily** Hours Saved With Optimization = **150 hours saved**
  - Total Workflow **Monthly** Hours Saved With Optimization = **4,500 hours saved**

Thread Zeppelin – Actionable Output

in Delta Lake

- **Thread level identifiers:** ingestion_date, thread_id, process_id, unique_app_id
- **Class/Function runtime:** class_name, function_name, execution_time, start_time, end_time
- **Resource consumption:** memory_usage, cpu_usage, recursion_limit
- **Attributes:** arguments, keyword_arguments, return_value
- **Memory-efficient object code:** source_code_compressed, source_code_md5_hash

```
code is running in databricks....
```

Table

	ingestion_date	thread_id	process_id	unique_app_id	class_name	function_name	execution_time	start_time	end_time	
10	2024-09-05 21:10:29	139905204221504	7028	xxxxxxxxxxxx	Packager	clean_bundle()	16.618201	2024-09-05 21:10:12.375579	2024-09-05 21:10:28.993780	0
11	2024-09-05 21:10:09	139907463598720	7028	xxxxxxxxxxxx	JobBundler	start_and_wait_bundle_jobs()	13.032840	2024-09-05 21:09:55.184393	2024-09-05 21:10:08.217233	18
12	2024-09-05 21:09:55	139907463598720	7028	xxxxxxxxxxxx	JobBundler	add_bundle()	12.573318	2024-09-05 21:09:41.607485	2024-09-05 21:09:54.180803	0
13	2024-09-05 21:14:21	139907463598720	7028	xxxxxxxxxxxx	Installer	load_demo_cluster()	12.079351	2024-09-05 21:14:08.445860	2024-09-05 21:14:20.525211	54
14	2024-09-05 21:17:12	139907463598720	7028	xxxxxxxxxxxx	InstallerReport	display_install_result()	7.020407	2024-09-05 21:17:04.927202	2024-09-05 21:17:11.947608	0
15	2024-09-05 21:09:50	139907463598720	7028	xxxxxxxxxxxx	JobBundler	reset_staging_repo()	6.430449	2024-09-05 21:09:42.609207	2024-09-05 21:09:49.039656	0
16	2024-09-05 21:14:01	139907463598720	7028	xxxxxxxxxxxx	Installer	get_demo_conf()	5.143853	2024-09-05 21:13:55.362172	2024-09-05 21:14:00.506025	0
17	2024-09-05 21:13:51	139907463598720	7028	xxxxxxxxxxxx	Installer	cluster_is_serverless()	5.096056	2024-09-05 21:13:45.016361	2024-09-05 21:13:50.112416	0
18	2024-09-05 21:10:02	139907463598720	7028	xxxxxxxxxxxx	JobBundler	create_or_update_bundle_jobs()	5.013854	2024-09-05 21:09:56.186044	2024-09-05 21:10:01.199898	18
19	2024-09-05 21:14:07	139907463598720	7028	xxxxxxxxxxxx	InstallerReport	display_install_info()	5.012233	2024-09-05 21:14:01.509456	2024-09-05 21:14:06.521688	0
20	2024-09-05 21:14:52	139907463598720	7028	xxxxxxxxxxxx	Installer	get_or_create_endpoint()	4.464724	2024-09-05 21:14:46.602878	2024-09-05 21:14:51.067602	0

Thread Zeppelin – Prescriptive Guidance

AI-assisted features

- ➔ Function source code is integrated with our very own DBRX
- ➔ Includes additional columns:
  - ◆ Function optimization **potential indicator (high, med, low)**: string
  - ◆ LLM **suggestions** for optimization techniques: list
  - ◆ LLM **optimized function code snippet**: string

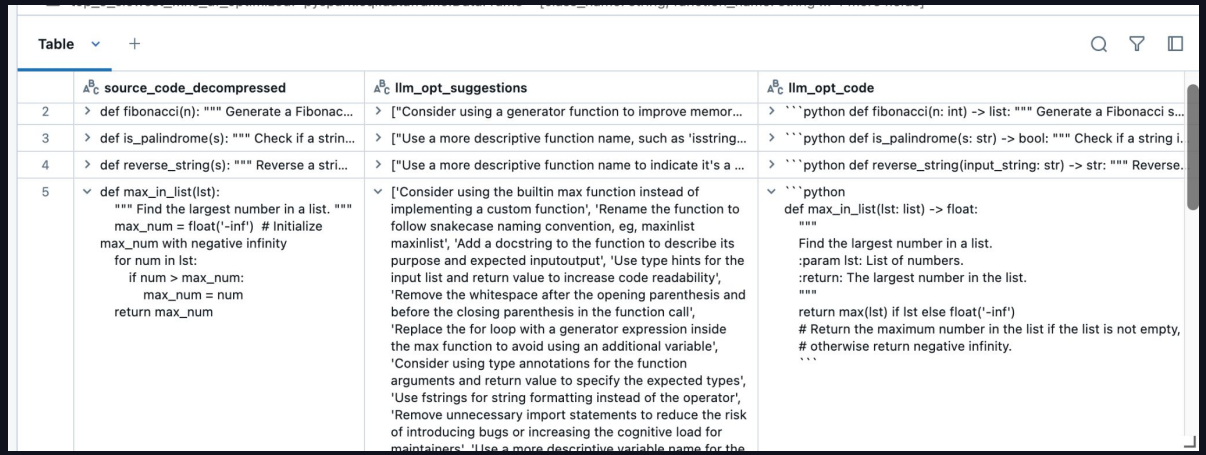


Table +

	source_code_decompressed	llm_opt_suggestions	llm_opt_code
2	def fibonacci(n): """ Generate a Fibonac...	["Consider using a generator function to improve memor...	```python def fibonacci(n: int) -> list: """ Generate a Fibonacci s...
3	def is_palindrome(s): """ Check if a strin...	["Use a more descriptive function name, such as 'isstring...	```python def is_palindrome(s: str) -> bool: """ Check if a string i...
4	def reverse_string(s): """ Reverse a stri...	["Use a more descriptive function name to indicate it's a ...	```python def reverse_string(input_string: str) -> str: """ Reverse...
5	def max_in_list(lst): """ Find the largest number in a list. """ max_num = float('-inf') # Initialize max_num with negative infinity for num in lst: if num > max_num: max_num = num return max_num	['Consider using the builtin max function instead of implementing a custom function', 'Rename the function to follow snakecase naming convention, eg, maxinlist maxinlist', 'Add a docstring to the function to describe its purpose and expected inputoutput', 'Use type hints for the input list and return value to increase code readability', 'Remove the whitespace after the opening parenthesis and before the closing parenthesis in the function call', 'Replace the for loop with a generator expression inside the max function to avoid using an additional variable', 'Consider using type annotations for the function arguments and return value to specify the expected types', 'Use fstrings for string formatting instead of the operator', 'Remove unnecessary import statements to reduce the risk of introducing bugs or increasing the cognitive load for maintainers', 'Use a more descriptive variable name for the	```python def max_in_list(lst: list) -> float: """ Find the largest number in a list. :param lst: List of numbers. :return: The largest number in the list. """ return max(lst) if lst else float('-inf') # Return the maximum number in the list if the list is not empty, # otherwise return negative infinity. ```

Thread Zeppelin – Expansion Opportunities

Enable customization and make more accessible to BI users

- Integrate function source code with Model Serving Endpoint & extend metrics
  - Include additional columns such as:
    - LLM optimized function code run status
    - Estimated runtime reduction for each optimized function/class
  - Swap out with another Foundation model or custom/fine-tuned model
  - Integrate with Databricks Assistant API & System Tables
- Integrate output Delta table into an AI/BI Genie Space and Dashboard providing:
  - Quick Analysis
  - Immediate Prescriptive Action
  - Additional DBSQL $DBU Consumption
- Package into SQL function with simple input parameters

Let’s watch THREAD ZEPPELIN in action!

Problem Statement

Analyzing large code bases for optimization opportunities can be a tedious and time-consuming process

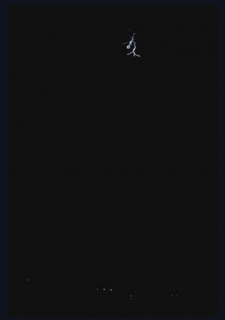
Existing Solutions

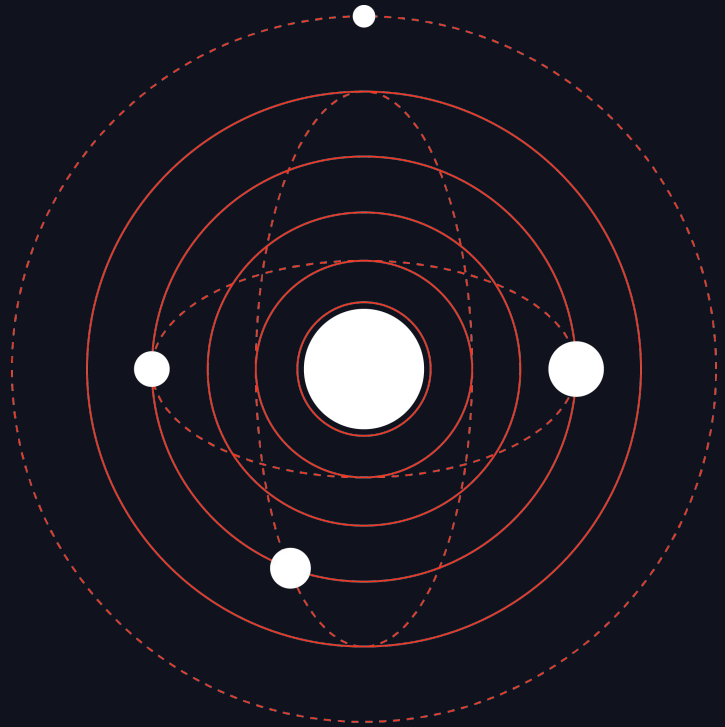
- Output runtime details line-by-line
  - cProfile
  - Scalene
  - Memory Profiler
  - Yappi
- Introduce significant overhead
- Produce more detail than can be quickly interpreted and actioned

Proposed Solution

- Profiles at a function level for much faster identification of problem code
- Leverages easy-to-use profiling decorator
- Can run single or multi-threaded
- Outputs results to a delta table, including a compressed version of the original source code

TECH SUMMIT





THREAD ZEPPELIN

An intelligent multi-threaded function-level code profiler

Presenter: Kayla Grieme, Sr. Solutions Architect

RSA Contributors: Robert Altmiller, Alan Reese, Divyank Jain