STAC Update:
Machine Learning

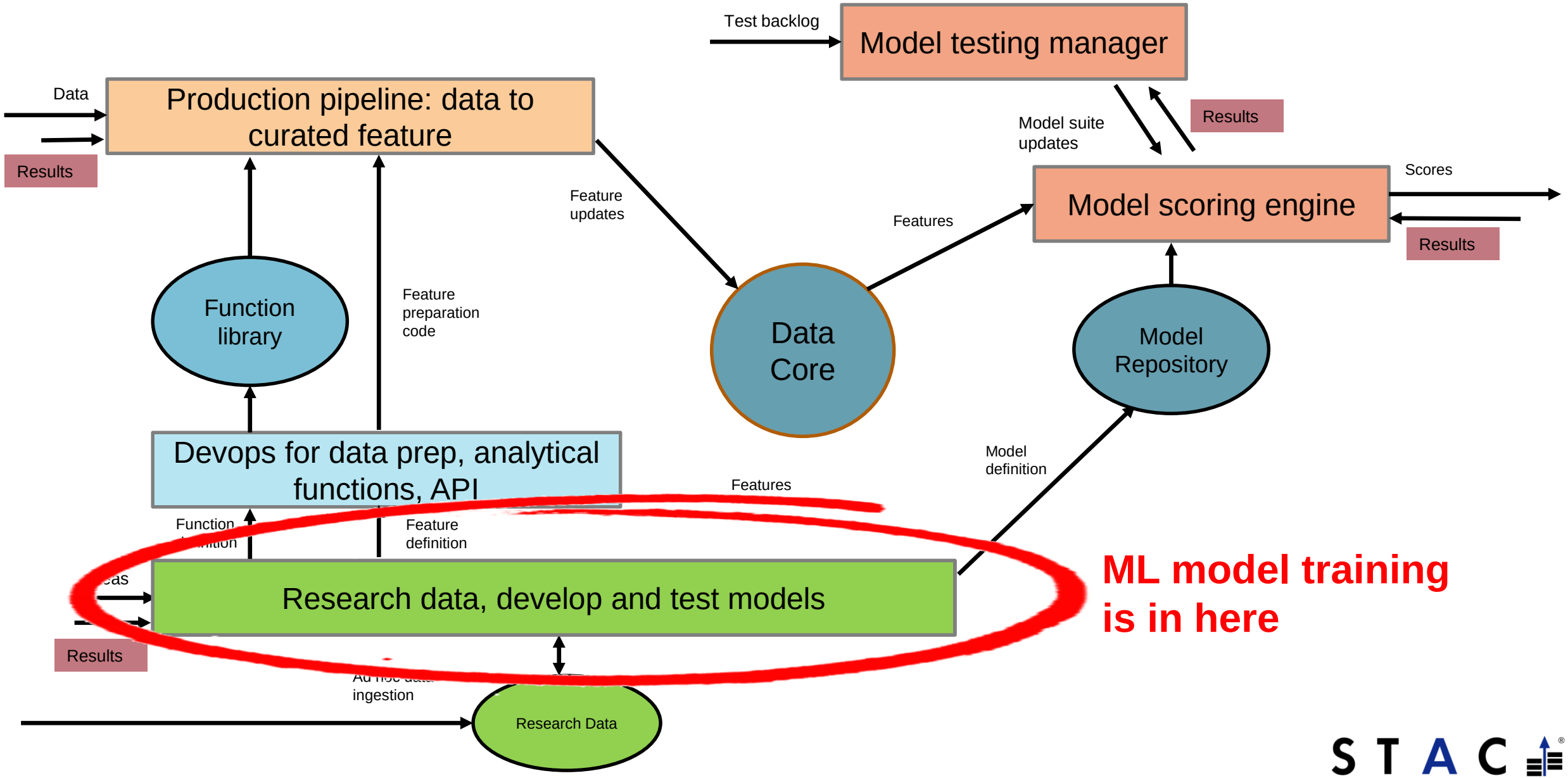
Michel Debiche,
Guest Analyst, STAC

michel.debiche@STACresearch.com

Background

- STAC is focused on helping with pain points in the data science process
- See QuantOps™ presentation from last STAC Summits
  - www.STACresearch.com/STAC-Summit-13-Jun-2018-debiche
- A key pain point: training of machine learning models

ML model training in context



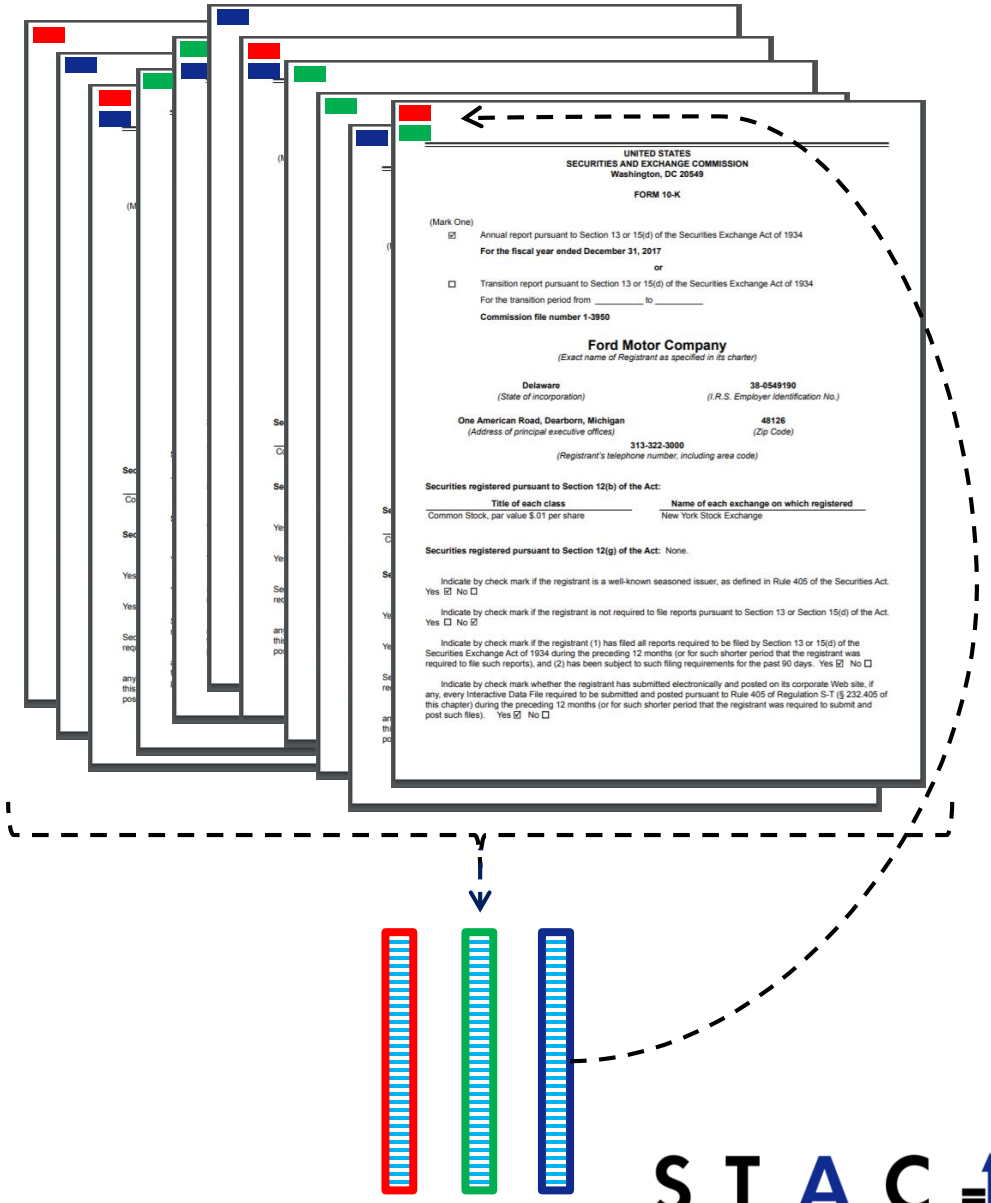
STAC
SECURITIES TECHNOLOGY ANALYSIS CENTER

Why we built a strawman benchmark for ML

- Many Council members have asked for ML benchmarks
- Getting a group to go from a blank sheet to usable benchmarks is hard
  - Many, many degrees of freedom
- We decided to propose something for feedback
  - This is not an official Council-backed benchmark suite
- Based on discussions with user firms and projects we've been involved in

Use case

- Topic modeling of form 10-K documents (from EDGAR) using Latent Dirichlet Allocation (LDA)
- Topic models identify and quantify similarities among documents
- This is unsupervised learning
- The engineering challenge is the training phase



STAC®
SECURITIES TECHNOLOGY ANALYSIS CENTER

Examples of top 10 words from 3 topics

- Topic #3: healthcare patient hospital physician reimbursement payor pharmacy clearance clinic nursing privacy
- Topic #5: paper fiber chemical print specialty packaging coating wood mill glass plastic
- Topic #7: gas oil pipeline drilling well transportation emission exploration drill commodity

Why we chose this use case

- ML on unstructured text is a wide category of use cases
- It's an active area of academic & industrial research
- ML on 10-K forms is common in finance
- LDA is popular with "quantamental" firms thanks to explainability
- Not a toy problem, but simple enough to constrain benchmark costs
- Several implementations (e.g., Scikit-Learn, Gensim, MLLib)
- Can work on many platforms (processors, frameworks, MLaaS, etc.)
- May be relatable to other document-similarity algos (e.g., word2vec)

Metrics

- Performance
  - How fast did the solution produce a model?
- Cost
  - What did it cost for the solution to produce a model?
- Quality
  - How good is the model the solution produced?

➢ If we allow all three to vary, how do we make comparisons?

Answer: Expose the tradeoffs of a each solution

STAC®
SECURITIES TECHNOLOGY ANALYSIS CENTER

What is quality?

- What ultimately matters is business utility
- No absolute right answer
- Hierarchy of quality metrics:
  - In-sample goodness-of-fit: stopping criteria for iterative algorithms
  - Out-of-sample goodness-of-fit: compare model results
  - Quality metric of some result derived from model (e.g. clusters)
  - Simulated business result (e.g. trading signal backtest)

How do you find the best quality?

- Usual process: Iterative search over lattice of parameter values
- Use consistent in-sample goodness-of-fit tolerance for each node
  - Perplexity improvement tolerance
- Also compute an out-of-sample goodness-of-fit metric to compare nodes
  - Perplexity of held-out documents (PHOD)

But there is a problem …

- PHOD on its own is inadequate for choosing answer, because:
  - Feature selection can affect average level of PHOD
  - It has poor correlation with subjective evaluation of topic quality
- For some parameter grids, minimum PHOD produces junk topics
  - Topic #22: •whether extinguish eyewear eyelid eyeglass eye exudate extrusion extruder extrude extremity extreme extravasation extrapolation extrapolate extranet extractor extraction extract extracorporeal
- Choices:
  - Apply better objective function to derived results (e.g. cluster coherence)
  - Human in the loop
- PHOD can still identify relevant subsets of parameter space

How do we compare systems?

- A proxy for achievable quality is number *N* of experiments run
- *N* can be increased by going faster or wider (parallelism)
- Grid search can be more efficient (i.e. fewer experiments to find same quality) by reevaluating grid between grid searches, and running more cycles
- Time to best quality can be reduced by running finer grid and more experiments per cycle
- What are the tradeoffs between cost and cycle time for different systems?

Initial tests

- Common to all stacks tested:
  - Scikit-Learn 0.20.0
  - SpaCy 2.0.12
  - RHEL 7
  - GCP
- Common to all workloads:
  - Quality (fixed the parameters to be searched)
- Varied:
  - Python distribution
  - Instance type

Findings: Python distributions

- Tested a single experiment on an n1-standard-16 instance
- First distro: Anaconda Python 3, MKL 2019
- Second distro: Intel Python 2018, Intel MKL 2018
- Non-Intel Python took 19% more time to complete
  - That is, it cost 19% more than Intel Python
- We standardized on Intel Python for the rest of the project

Findings: Parallelism

- Some common tenets regarding cloud and parallel processing have caveats, especially for this use case.
- Seemingly obvious or default choices for parameters of our software stack lead to counter-intuitive results.
- Details in following slides.

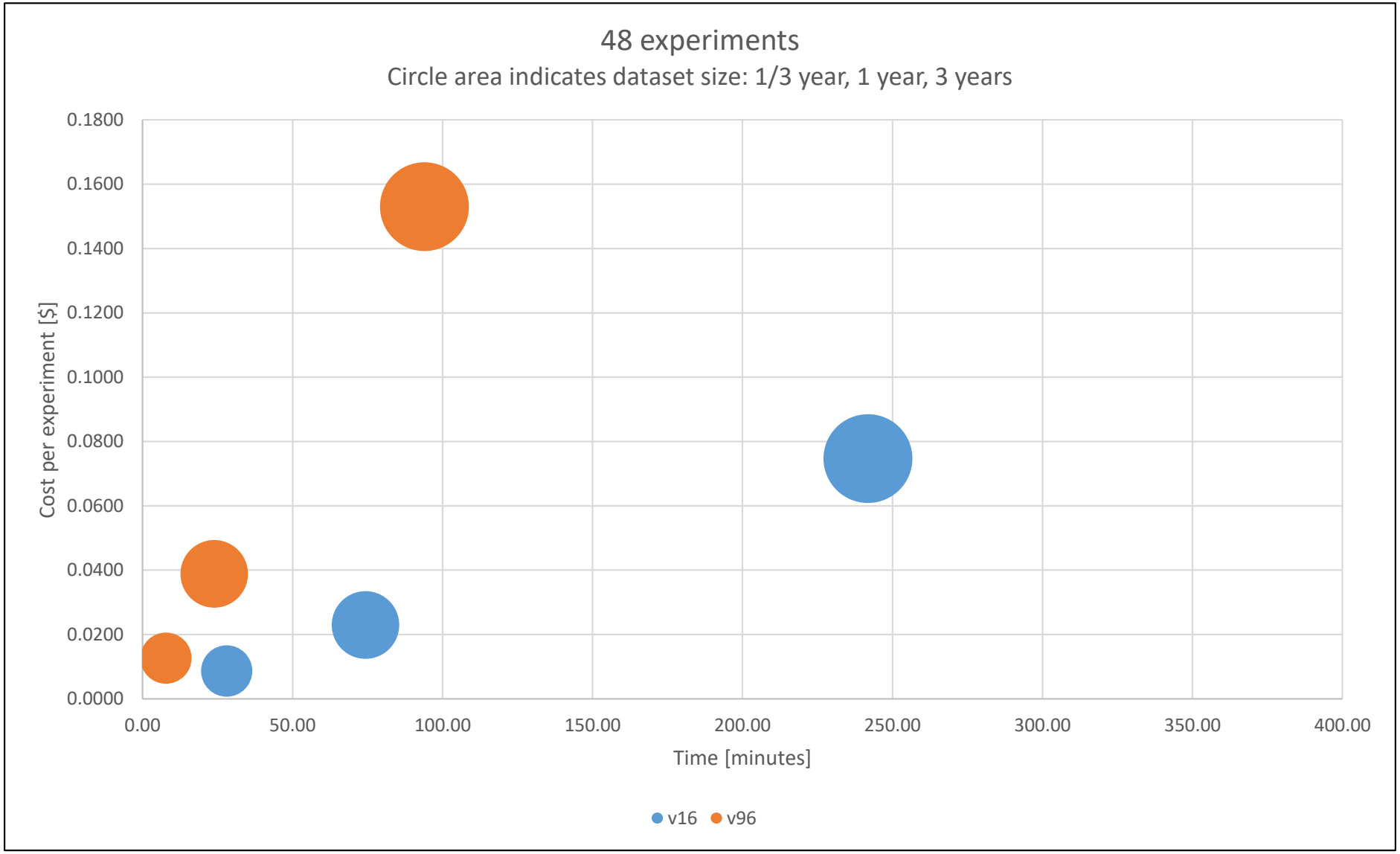
Findings: Threading

- *MKL_NUM_THREADS* parameter of MKL library faithfully utilizes nearly 100% of all physical cores on a platform
- Unfortunately, the overhead with this LDA implementation exceeded the benefit
- Best performance was a single thread per experiment
- So set *MKL_NUM_THREADS* environment *variable to 1.*

Findings: Multi-processing

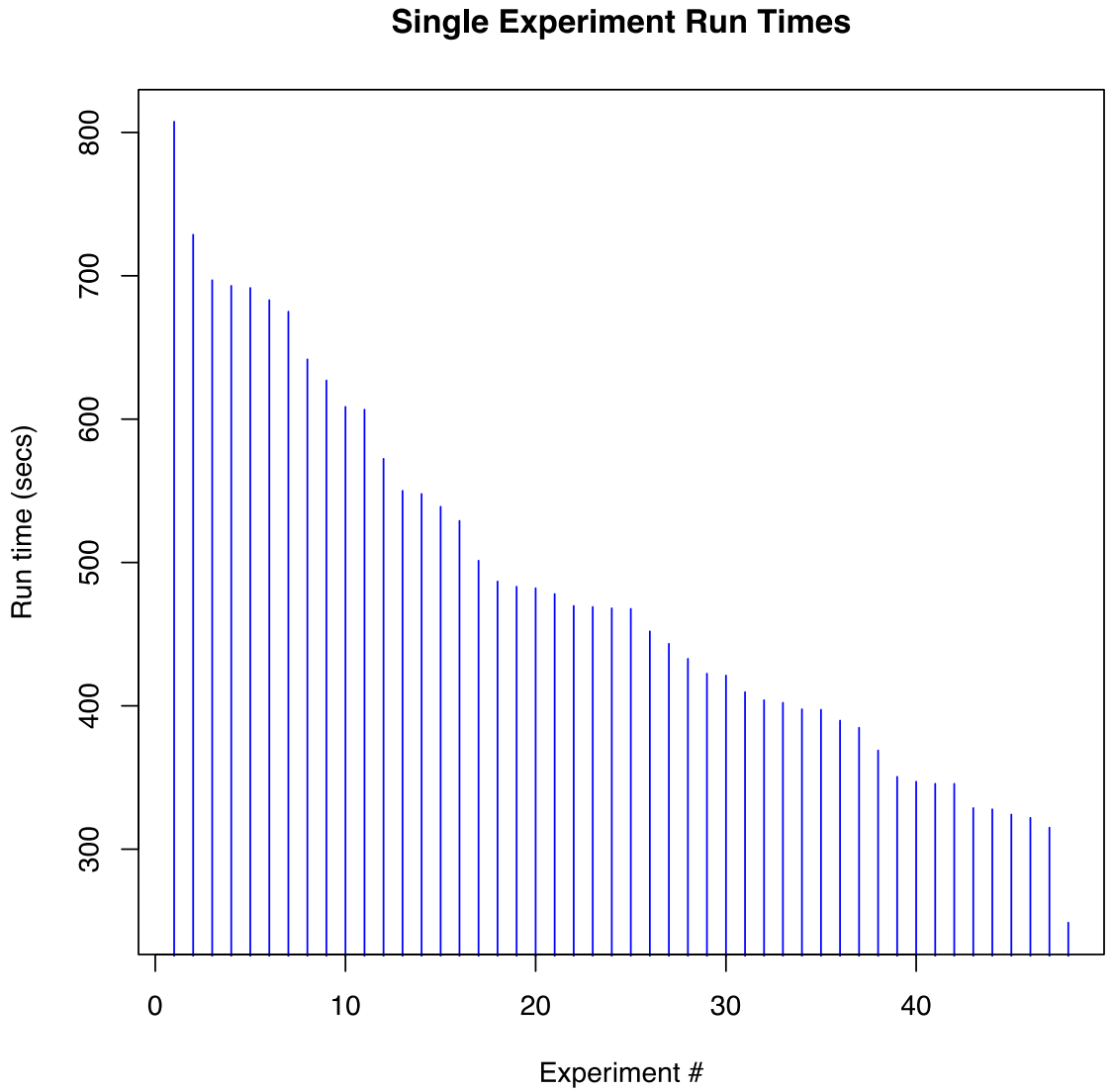
- *n-jobs* parameter of Scikit-Learn LDA model uses multiple processes to parallelize the separable part of the algorithm
- Unfortunately, the overhead exceeded the benefit
- If combined with default MKL setting, contention for cores made performance even worse
- So, set *n_jobs* parameter to 1

Findings: Scaling with parallel processes



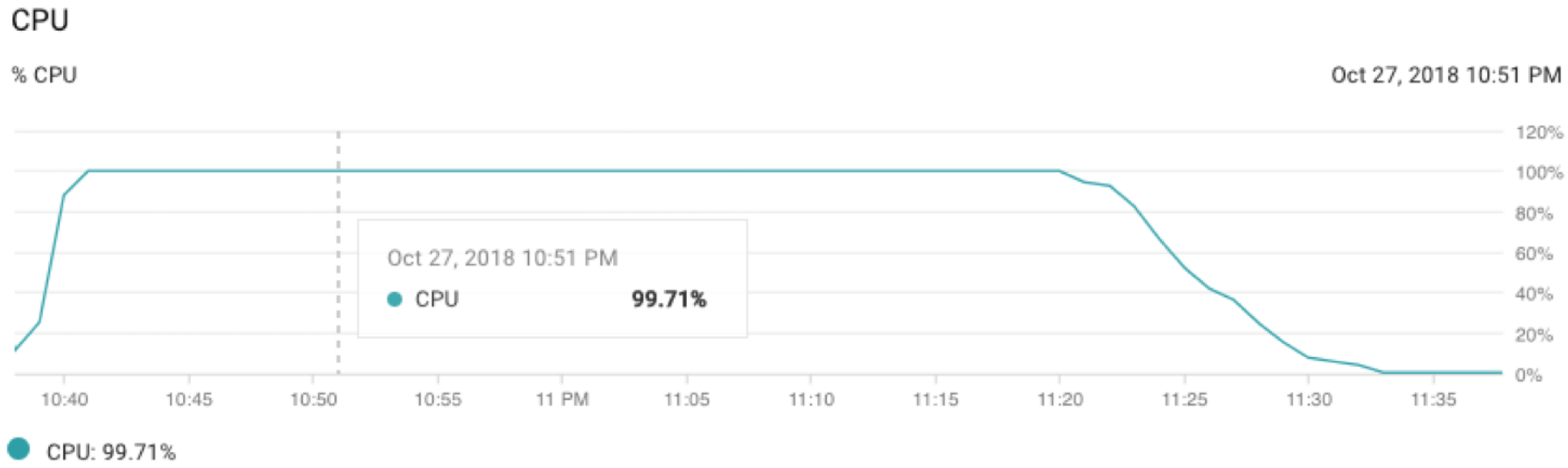
STAC®
SECURITIES TECHNOLOGY ANALYSIS CENTER

Distribution of experiment run times



STAC®
SECURITIES TECHNOLOGY ANALYSIS CENTER

CPU utilization with unequal run-times (v16)



STAC®
SECURITIES TECHNOLOGY ANALYSIS CENTER

Findings: Scaling with parallel processes (3 year dataset)

- n1-standard-16: 28.01 minutes
  - ~ 23,000 vCPU seconds
- n1-standard-96: 7.77 minutes
  - ~ 54,000 vCPU seconds
- 2.6x speedup from 6x the vCPUs; 2x the cost.
- If your goal is to minimize IT operating cost, then n1-standard-16 wins
- But if your goal is to maximize data scientist productivity, then n1-standard-96 wins
  - <10 minutes generally thought short enough to allow interactive exploration

Key findings

- Intel Python seems to offer higher speed/lower cost with this workload
- Scikit-Learn LDA works best with one thread per experiment
- Process-level parallelism does not scale linearly with vCPUs
- Scaling up is still desirable if cycle time (hence data scientist productivity) is at a premium
- Low-level objective quality measure PHOD alone is inadequate

Next steps

- Look at scale out vs scale up
- Write up initial project
  - Next few weeks
- Form working group
  - Provide feedback on this use case
  - Propose other key use cases
- Test more implementations, more platforms