Coiled
A Dask Company

dask

Observability for Distributed Computing with Dask

How to remain sane while identifying and solving your problems.

Hendrik Makait
@hendrikmakait
hendrik@coiled.io

dask

General parallelism for Python

- Parallel for-loopy code
- Dataframes / ETL
- Array computing
- Dynamic and complex systems / ML

dask

General parallelism for Python

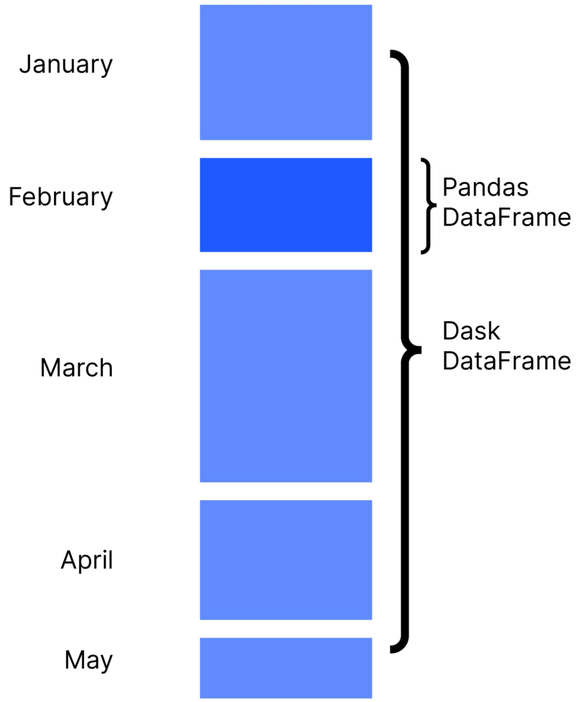
- Parallel for-loopy code
- Dataframes / ETL
- Array computing
- Dynamic and complex systems / ML

```
futures = []
for filename in filenames:
    future = client.submit(
        process,
        filename,
    )
    futures.append(future)
```

dask

General parallelism for Python

- Parallel for-loopy code
- Dataframes / ETL
- Array computing
- Dynamic and complex systems / ML



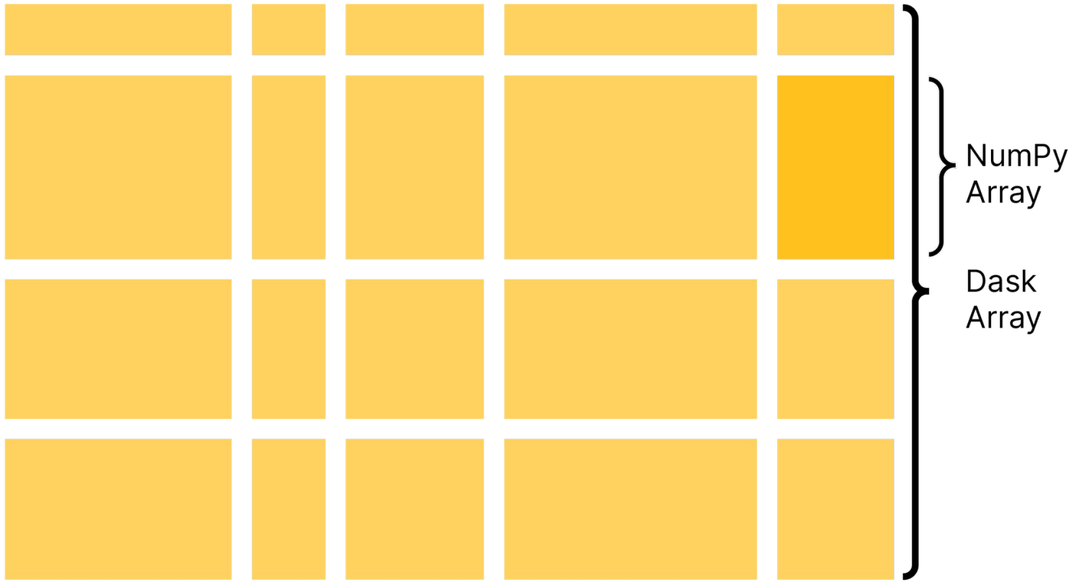

```
import dask.dataframe as dd

df = dd.read_csv("s3://…/*.csv")
…
df.to_parquet("s3://…/clean.parquet")
```

dask

General parallelism for Python

- Parallel for-loopy code
- Dataframes / ETL
- Array computing
- Dynamic and complex systems / ML



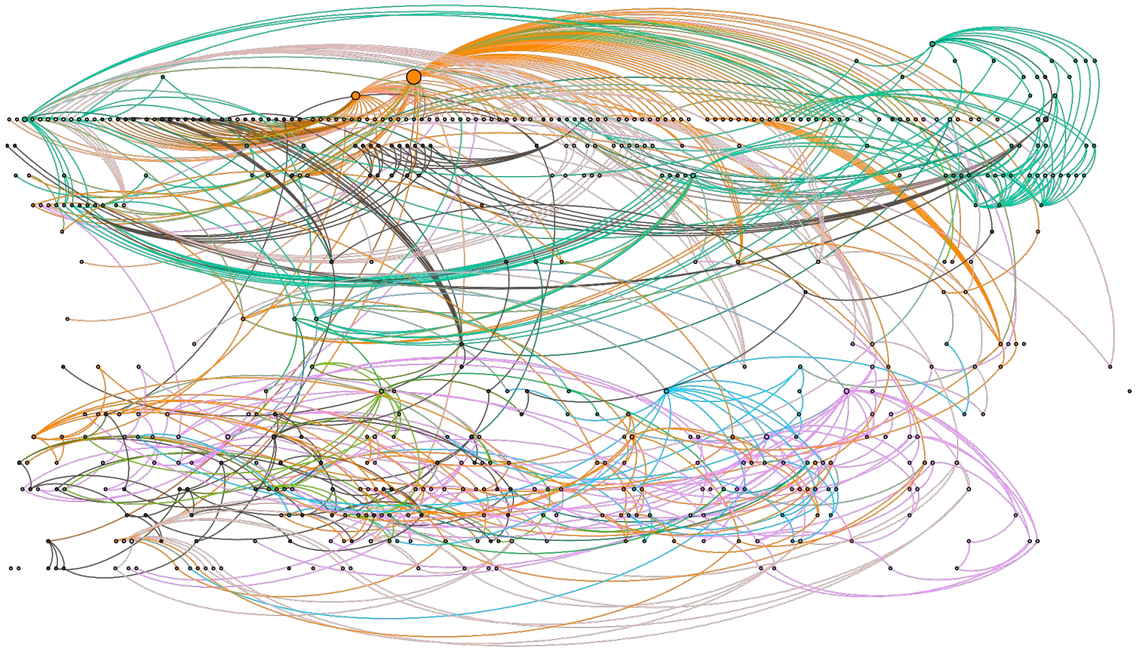

```
import dask.array as da

x = da.from_array(...)

x -= x.mean(axis=2)
x.dot(x.T).sum(mean=0)
```

dask

General parallelism for Python



- Parallel for-loopy code
- Dataframes / ETL
- Array computing
- Dynamic and complex systems / ML

```
import prefect
import xgboost.dask
import lightgbm
import optuna
import joblib
import xarray
...
```

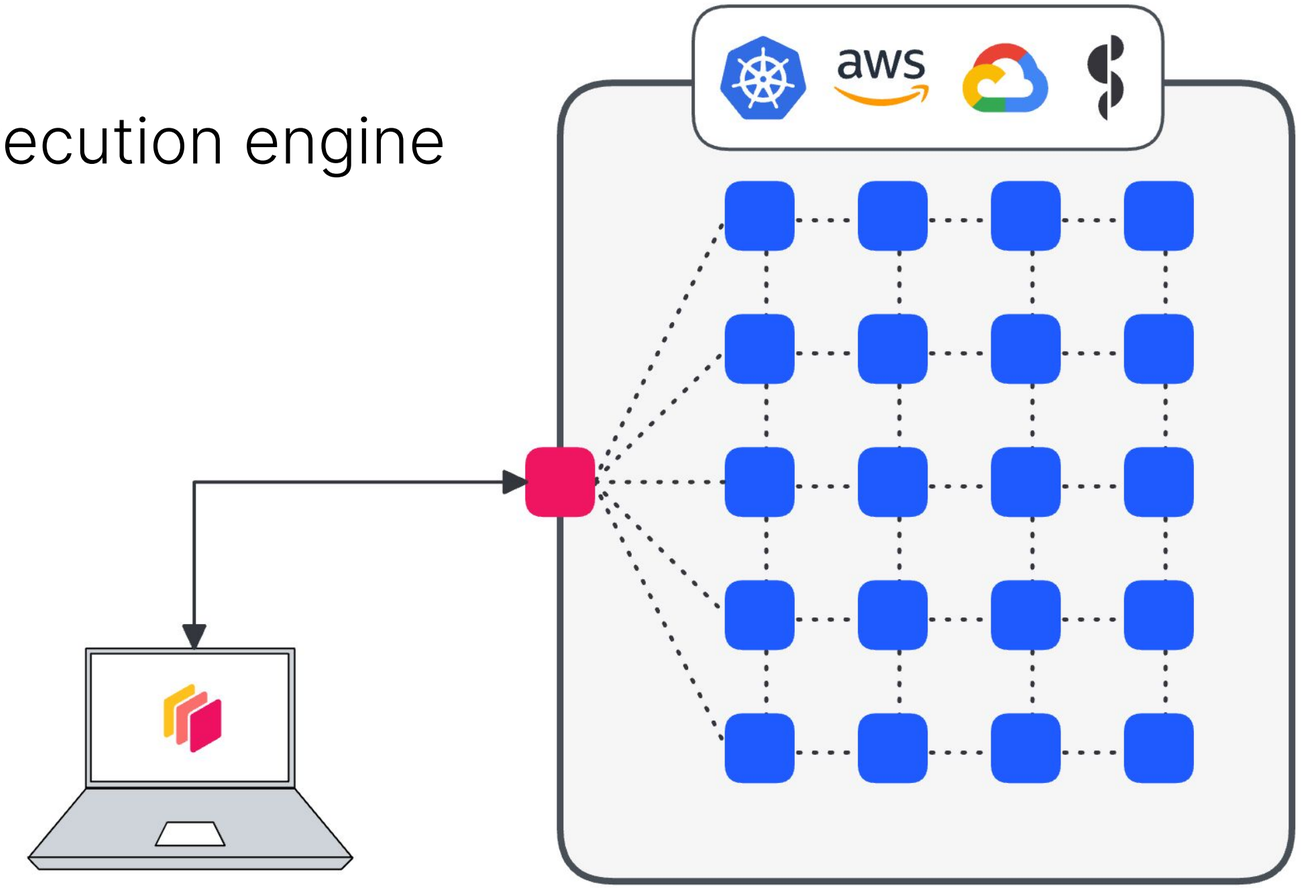
dask

Distributed computing for Python

Scales Python with its distributed execution engine

Several deployment options:

- Kubernetes
- HPC
- Cloud
- Coiled



Observability

Observability

What is Observability?

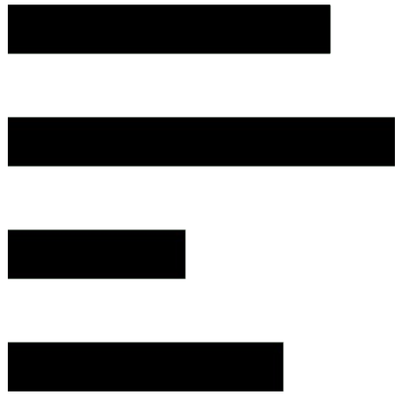
“Observability lets us **understand a system from the outside**, by letting us ask questions about that system **without knowing its inner workings**. Furthermore, it allows us to easily troubleshoot and handle novel problems (i.e. “unknown unknowns”), and helps us answer the question, **“Why is this happening?”**

– https://opentelemetry.io/docs/concepts/observability-primer/

→ An observable system enables us to monitor and debug it.

The pillars of observability

Logging



Metrics

Tracing

Diagnostic Tooling

Logging

```
2023-03-20 15:14:43,852 - distributed.nanny - INFO -         Start Nanny at: 'tls://10.244.10.11:33815'
2023-03-20 15:14:46,216 - distributed.worker - INFO -       Start worker at:   tls://10.244.10.11:46721
2023-03-20 15:14:46,216 - distributed.worker - INFO -          Listening to:   tls://10.244.10.11:46721
2023-03-20 15:14:46,216 - distributed.worker - INFO -           Worker name: dask-worker-351d74e2f7f644beb571a17408e81902-8z6lj
2023-03-20 15:14:46,216 - distributed.worker - INFO -          dashboard at:          10.244.10.11:8787
2023-03-20 15:14:46,216 - distributed.worker - INFO - Waiting to connect to: tls://dask-351d74e2f7f644beb571a17408e81902.staging:8786
2023-03-20 15:14:46,216 - distributed.worker - INFO - -------------------------------------------------
2023-03-20 15:14:46,216 - distributed.worker - INFO -               Threads:                          1
2023-03-20 15:14:46,217 - distributed.worker - INFO -                Memory:                   8.00 GiB
2023-03-20 15:14:46,217 - distributed.worker - INFO -       Local Directory: /tmp/dask-worker-space/worker-4j9p24_r
2023-03-20 15:14:46,217 - distributed.worker - INFO - -------------------------------------------------
2023-03-20 15:14:46,235 - distributed.worker - INFO -         Registered to: tls://dask-351d74e2f7f644beb571a17408e81902.staging:8786
2023-03-20 15:14:46,235 - distributed.worker - INFO - -------------------------------------------------
2023-03-20 15:14:46,236 - distributed.core - INFO - Starting established connection to tls://dask-351d74e2f7f644beb571a17408e81902.staging:8786
2023-03-20 15:14:59,341 - distributed.core - INFO - Event loop was unresponsive in Worker for 9.18s.  This is often caused by long-running GIL-holding functions
or moving large chunks of data. This can cause timeouts and instability.
2023-03-20 15:15:00,410 - distributed.utils_perf - INFO - full garbage collection released 47.80 MiB from 0 reference cycles (threshold: 9.54 MiB)
2023-03-20 15:15:03,907 - distributed.core - INFO - Event loop was unresponsive in Worker for 4.53s.  This is often caused by long-running GIL-holding functions
or moving large chunks of data. This can cause timeouts and instability.
2023-03-20 15:15:43,799 - distributed.core - INFO - Event loop was unresponsive in Worker for 3.59s.  This is often caused by long-running GIL-holding functions
or moving large chunks of data. This can cause timeouts and instability.
2023-03-20 15:15:48,464 - distributed.core - INFO - Event loop was unresponsive in Worker for 4.66s.  This is often caused by long-running GIL-holding functions
or moving large chunks of data. This can cause timeouts and instability.
2023-03-20 15:15:51,847 - distributed.core - INFO - Event loop was unresponsive in Worker for 3.38s.  This is often caused by long-running GIL-holding functions
or moving large chunks of data. This can cause timeouts and instability.
2023-03-20 15:16:29,763 - distributed.comm.tcp - INFO - Connection from tls://10.244.10.9:44336 closed before handshake completed
2023-03-20 15:18:05,967 - distributed.shuffle._comms - ERROR - Shuffle cc9531b7820cc766da633788babf811c forgotten
Traceback (most recent call last):
  File "/srv/conda/envs/notebook/lib/python3.10/site-packages/distributed/shuffle/_comms.py", line 71, in _process
    await self.send(address, shards)
  File "/srv/conda/envs/notebook/lib/python3.10/site-packages/distributed/shuffle/_worker_extension.py", line 122, in send
    self.raise_if_closed()
  File "/srv/conda/envs/notebook/lib/python3.10/site-packages/distributed/shuffle/_worker_extension.py", line 163, in raise_if_closed
    raise self._exception
RuntimeError: Shuffle cc9531b7820cc766da633788babf811c forgotten
2023-03-20 15:18:05,977 - distributed.shuffle._comms - ERROR - Shuffle cc9531b7820cc766da633788babf811c forgotten
Traceback (most recent call last):
  File "/srv/conda/envs/notebook/lib/python3.10/site-packages/distributed/shuffle/_comms.py", line 71, in _process
    await self.send(address, shards)
```

Logging best practices

Quick review: How to do good logging?

1. Don’t use **`print()`**.
2. Provide the necessary context.
3. Use structured logging.

Distributed logging

How to access all your logs in a distributed environment?

```
>>> cluster.scale(100)
```

Congratulations, you now have 102 different places to look for logs!

Distributed logging

How can I access all my logs in a distributed environment?

Built-in:

```
>>> client.get_scheduler_logs()

>>> client.get_worker_logs(workers=["10.0.2.96"])
```

Problem: If it's dead, it doesn't return logs.

Centralized logging

How can I access all my logs in a distributed environment?

Better: Centralized logging in an external system

- Persistence
- Querying

```
$ coiled cluster logs --cluster 183522 --workers 10.0.2.96
    --filter "memory usage"

(10.0.2.96)     2023-04-03 00:28:30.454000 distributed.worker.memory - WARNING - Worker is at 80%
memory usage. Pausing worker.  Process memory: 5.76 GiB -- Worker memory limit: 7.15 GiB
(10.0.2.96)     2023-04-03 00:28:31.657000 distributed.worker.memory - WARNING - Worker is at 79%
memory usage. Resuming worker. Process memory: 5.70 GiB -- Worker memory limit: 7.15 GiB
```

Metrics

What are metrics?

-

Metrics of a Dask cluster

`distributed` provides detailed statistics about its parts.

dask_scheduler_workers
Number of workers known by scheduler

dask_worker_memory_bytes
Memory breakdown per worker

...and many others.

See the full list at https://distributed.dask.org/en/latest/prometheus.html.

Centralized metrics

Enabling automated monitoring and diagnostics on the entire cluster

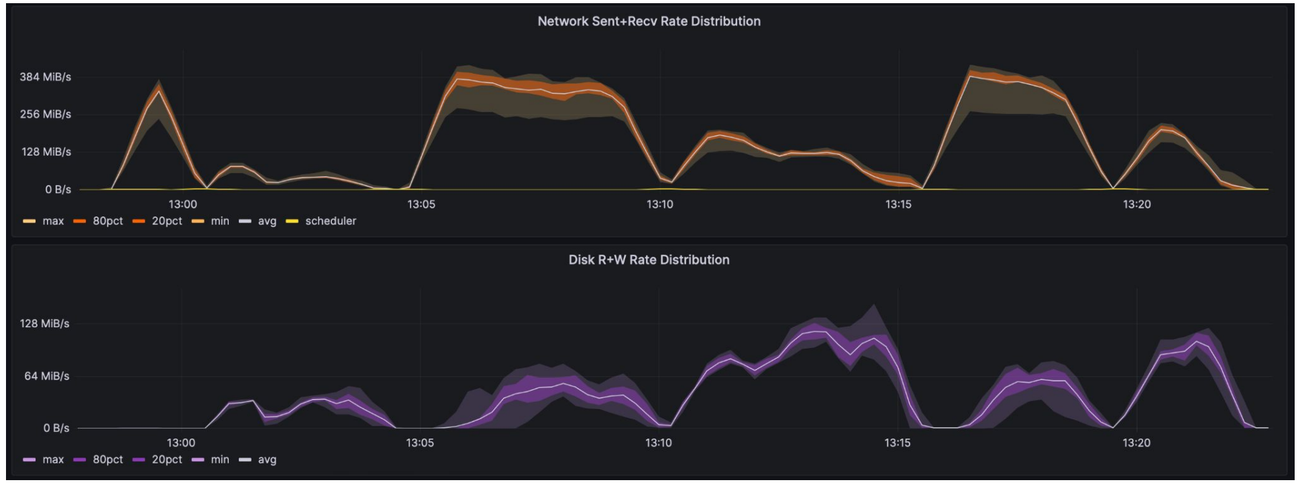
- Dask has a native Prometheus integration
- Centralized metrics enable:
  - Powerful dashboards

    https://benchmarks-grafana.oss.coiledhq.com/
  - Metric-based alerts/flags

Use Case

Choosing the best hardware

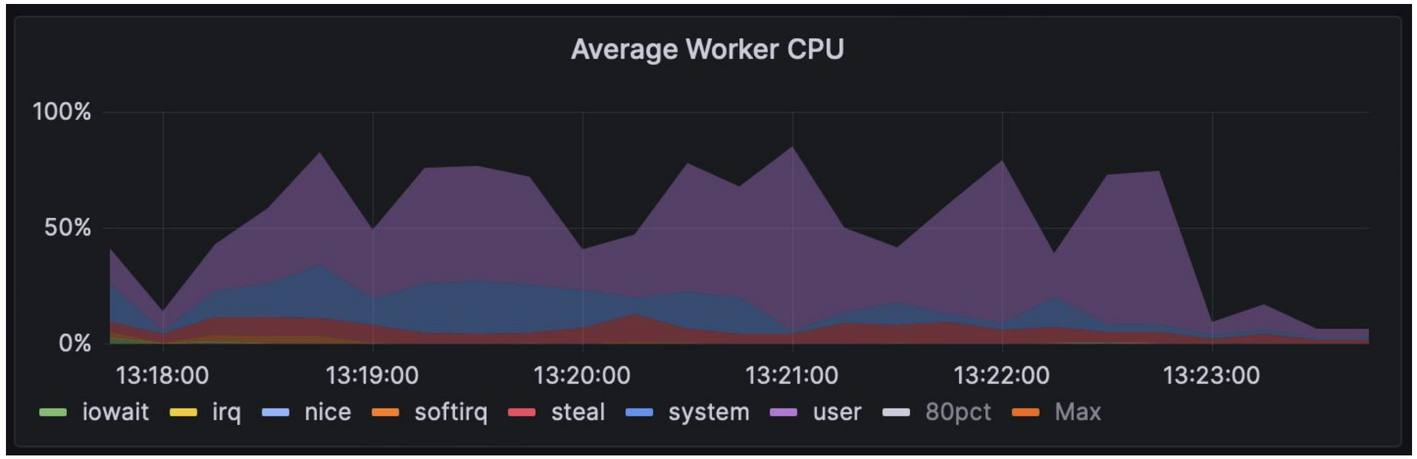
Workload 1: Network and disk with array workload



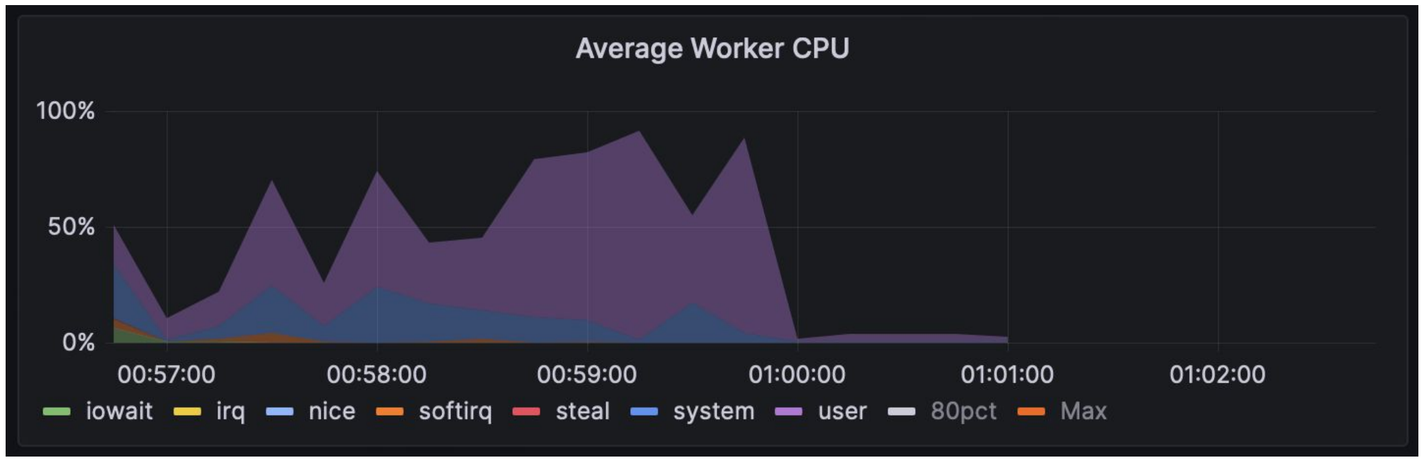
t3.large

m6i.large

Workload 2: XGBoost + Optuna with moderate CPU use



t3.large



m6i.large

Use case: Choosing the best hardware

Which AWS instance types should we choose as default?

Result: For data science/engineering workloads, non-burstable instances are both cheaper per hour and faster.

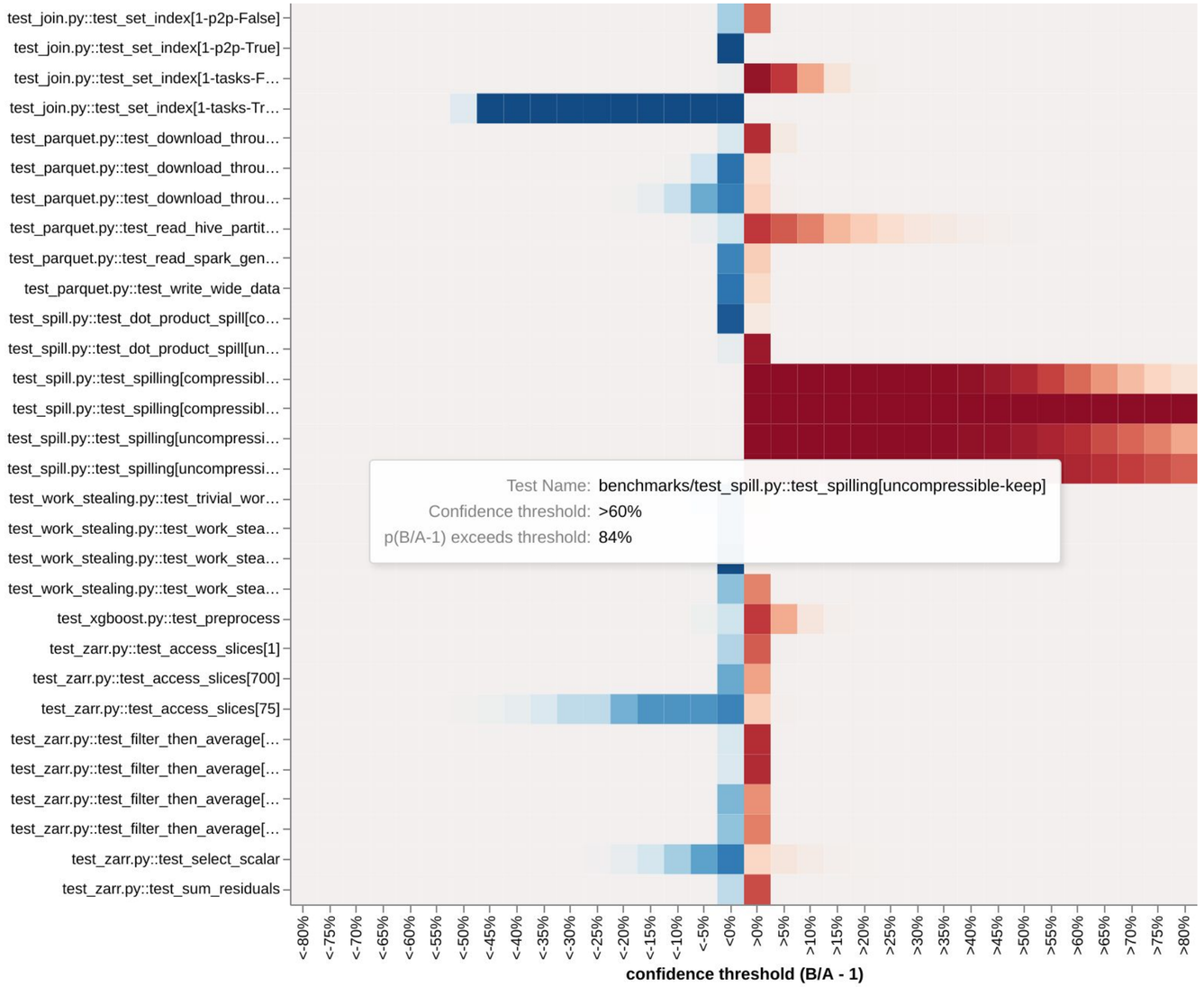
Find the full blog post at https://blog.coiled.io/blog/burstable-vs-nonburstable.

Use case: Data-driven development

Data-driven design for the Dask scheduler

Learn more at:

Data-driven design for the Dask scheduler
by *Guido Imperiale*
today, 14:10 - 14:40
in Hall B07-B08



Diagnostic Tooling

Dashboards

Tracking your clusters and your workloads.

Dashboards are

- 👍 **very useful**
- 👎 **ephemeral**

...what if we could store a subset of that data?

Performance reports

A built-in way for storing and sharing performance data about your workload execution

```
from distributed import performance_report

with performance_report(filename="pycon_de.html"):
    ...
```

Let’s take a look at an example performance report here.

Profiling

Gaining deep insights into performance

Built-in: **`/profile`**

- Statistical CPU profiler

👍 Low overhead → always-on

👍 Cluster aggregates or individual profiles

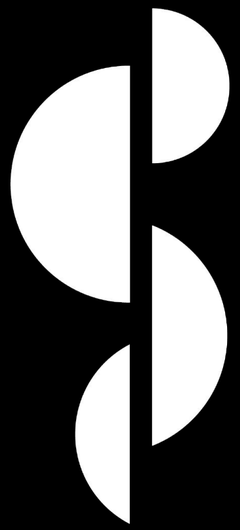
👎 Python-only

👎 Lacks rich timeline

For more in-depth profiling: **`dask-pyspy`**

Zooming back out

- **An observable system lets us ask: Why is this happening?**
- Logging and metrics are key pillars of an observable system.
- Good diagnostic tools refine the telemetry data to create insights.
- Dask has built-in logging, metrics, and diagnostic tools.
- Centrally and durably collect your telemetry data to fully leverage them.



Coiled

A Dask Company

Reliably scale Python in your cloud account. Get started for free.

coiled.io