

TaPS: A Performance Evaluation Suite for Task-based Execution Frameworks

J. Gregory Pauloski,^{*†} Valerie Hayot-Sasson,^{*†} Maxime Gonthier,^{*†} Nathaniel Hudson,^{*†} Haochen Pan,^{*} Sicheng Zhou,^{*} Ian Foster,^{*†} and Kyle Chard^{*}

^{}University of Chicago, [†]Argonne National Laboratory*

27 September 2024 — Chicago, IL

Modern Science Applications are *Task-centric*

- Applications **automate** computational processes to achieve a **scientific goal**
 - ◆ Designed as compositions of **discrete tasks**
 - ◆ Built/executed using a **task executor** (like Parsl or Globus Compute)
- **Diverse ecosystem** of task executor frameworks
 - ◆ Each with different **strengths** and **weaknesses**
 - ◆ Many **open challenges** to work on

Task Execution Frameworks

Manage the execution of tasks in parallel across arbitrary hardware.

Workflow Management Systems

Define, manage, and execute workflows represented by a directed acyclic graph (DAG) of tasks

~~Explicit~~

DAG defined via configuration file or domain specific language



Implicit

Task dependencies derived through dynamic evaluation of a procedural script



Concurrent Executors

On-demand asynchronous execution of tasks



How do we **benchmark** and compare *Python* execution frameworks?

The Status Quo

Ad Hoc Benchmarks

- Framework-specific examples/demos
- Custom, single-use evaluation scripts for a publication
- Forks of real science applications



Problems

- Code is **framework-specific**
- Ad-hoc scripts subject to **code rot**
- Porting applications can be **onerous**
- Subtle **errors** in ported applications can lead to **inaccurate comparisons**

SimGrid: a Generic
Large-Scale Distribut

Developing accurate and scalable sim
management systems with WRENCH

WfCommons: A Framework for Enabling Scientific Workflow Research and Development

Tainā Coleman^{a,c,*}, Henri Casanova^b, Loïc Pottier^a, Manav Kaushik^c, Ewa Deelman^{a,c}, Rafael Ferreira da Silva^{a,c,*}

Application skeletons: Construction and use in

Daniel S. Katz^{a,*}, Andre Merzky^b, Zhao Zhang^c, Shantenu Jha^b

^a Computation Institute, University of Chicago & Argonne National Laboratory, Chicago, IL, USA

^b RADICAL Laboratory, Rutgers

^c AMPLab, University of Califor

WfBench: Automated Generation of
Scientific Workflow Benchmarks

Tainā Coleman^{a,c,*}, Henri Casanova[†], Ketan Maheshwari[‡], Loïc Pottier^{*}, Sean R. Wilkinson[‡]

Justin Wozniak[§], Frédéric Suter[‡], Mallikarjun Shankar[‡], Rafael Ferreira da Silva[‡]

[†] University of Hawaii, Honolulu, HI, USA
[‡] National Laboratory, Oak Ridge, TN, USA

Prior work focused on **simulations** and **synthetic workloads**

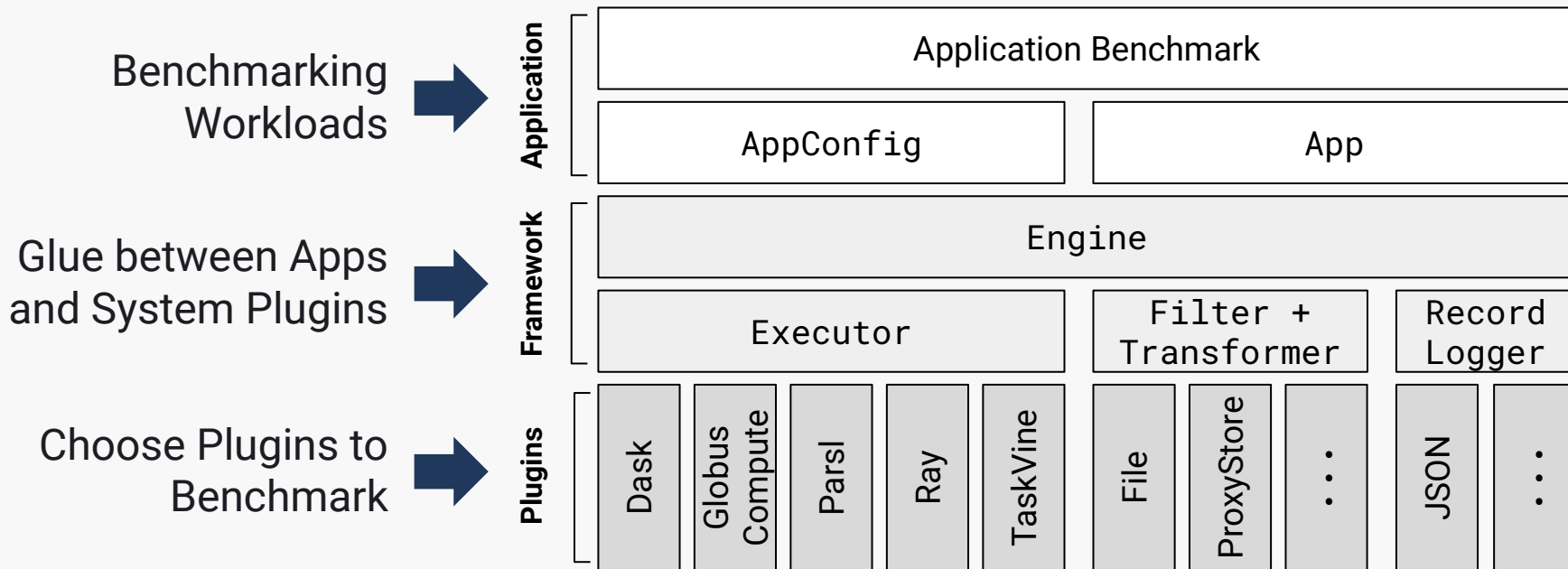
TaPS: Task Performance Suite

A standardized framework for evaluating task execution frameworks with real scientific workloads

Goals

- Provide reference/standard applications for benchmarking workloads
- Benchmark task executors / data management systems
- Robust and reproducible configuration system

Architecture

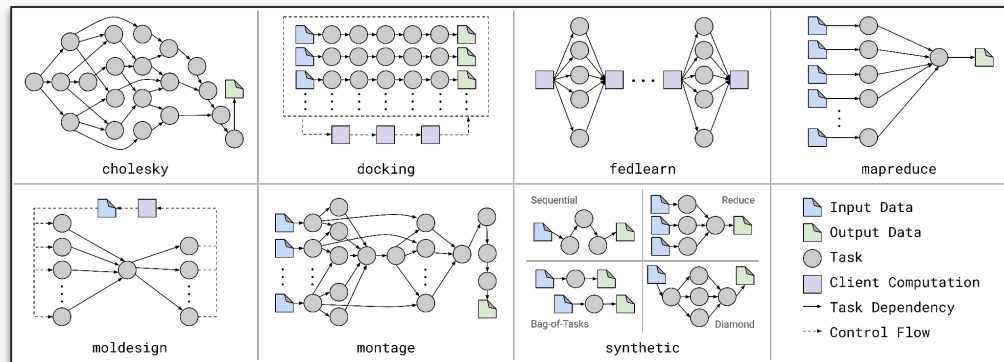


<https://taps.proxystore.dev/latest/api/>

Applications: *Benchmarking Workloads*

- Six Real Apps
- Two Synthetic
- Diverse Patterns
- Diverse Domains
- Per-App Guides
- Add your own!

Type	Name	Domain	Task Type(s)	Data Type(s)
<i>Real</i>	cholesky	Linear Algebra	Python	In-memory
	docking	Drug Discovery	Executable, Python	File
	fedlearn	Machine Learning	Python	In-memory
	mapreduce	Text Analysis	Python	File, In-memory
	moldesign	Molecular Design	Python	In-memory
	montage	Astronomy	Executable	File
<i>Synthetic</i>	synthetic	—	Python	In-memory
	failures	—	<i>Depends on base app</i>	<i>Depends on base app</i>



<https://taps.proxystore.dev/latest/apps/>

Engine Plugins: *Things You Can Compare*

	Task Execution	Data Management
Purpose	Asynchronously execute functions	Manage task data by filtering and transforming data into/resolve data from intermediate representations
Interfaces	Executor*	Filter + Transformer
Out-of-the-Box Implementations	ThreadPool, ProcessPool, Dask, Globus Compute, Parsl, Ray, TaskVine	Shared File Systems ProxyStore (DAOS, Globus Transfer, Margo, Redis, UCX, ZMQ, ...)

Check out taps.proxystore.dev/latest/guides to add new apps/plugins!

*Requires implicit data flow support via futures. Wrapper provided for implementations that lack this feature.

Using TaPS

Execute benchmarks with CLI or programmatically via API

```
$ python -m taps.run \  
  --app cholesky --app.matrix-size 10000 --app.block-size 1000 \  
  --engine.executor parsl-local --engine.executor.workers 16 \  
  --engine.transformer proxystore {transformer options} \  
  --engine.filter object-size {filter options} \  
  ...  
  
[Output Truncated]  
RUN (taps.run) :: Runtime directory: runs/cholesky-dask-2024-09-19-12-00-00  
APP (taps.apps.cholesky) :: Generated input matrix: (10000, 10000)  
APP (taps.apps.cholesky) :: Block size: 1000  
APP (taps.apps.cholesky) :: Output matrix: (10000, 10000)  
RUN (taps.run) :: Finished app (name=cholesky, runtime=13.18s)
```

Run directory:

- Logs for analysis
- Config for reproducibility
- Application outputs

```
+ runs  
├─ cholesky-dask-2024-09-19-11-00-00  
└─ cholesky-dask-2024-09-19-12-00-00  
   ├─ config.toml  
   ├─ log.txt  
   └─ tasks.jsonl
```

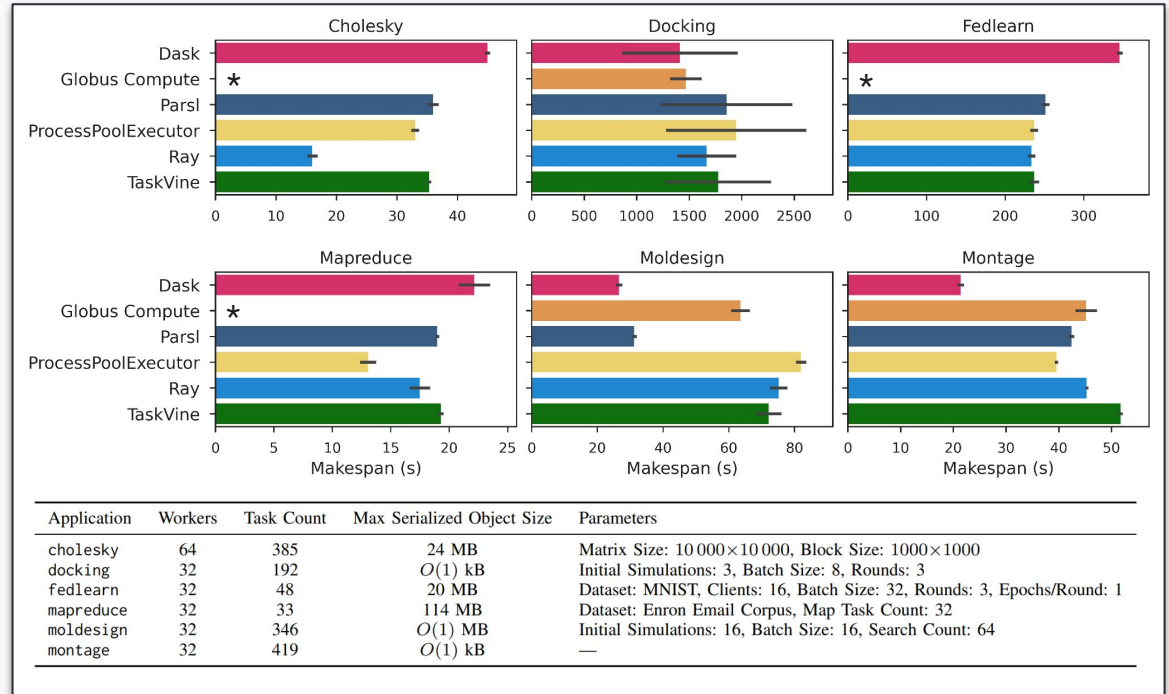
```
[app]  
name = "cholesky"  
matrix_size = 10000  
block_size = 1000  
  
[engine.executor]  
name = "dask"  
workers = 16  
  
[engine.filter]  
name = "object-size"  
min_size: 1000  
  
[engine.transformer]  
name = "proxystore"  
cache_size = 16  
extract_target = true  
populate_target = true  
...  
  
[logging]  
level = "INFO"  
file_level = "INFO"  
file_name = "log.txt"  
  
[run]  
dir_format = "runs/{name}-{executor}-{timestamp}"
```

<https://taps.proxystore.dev/latest/guides/config/>

```
python -m taps.run --config config.toml
```

Application Makespan

- No stand-out executor
- Diverse characteristics across ref. applications
- Highlight strengths and weaknesses across executor frameworks
- High-level investigation raises interesting questions



<https://github.com/proxystore/escience24-taps-analysis>

*Task data exceeds Globus Compute 10 MB payload limit.

Read the Paper!

TAPS: A Performance Evaluation Suite for Task-based Execution Frameworks

J. Gregory Pauloski,^{*†} Valerie Hayot-Sasson,^{*†} Maxime Gonthier,^{*†} Nathaniel Hudson,^{*†}
Haochen Pan,^{*} Sicheng Zhou,^{*} Ian Foster,^{*†} and Kyle Chard^{*†}

^{*}Department of Computer Science, University of Chicago, Chicago, IL, USA

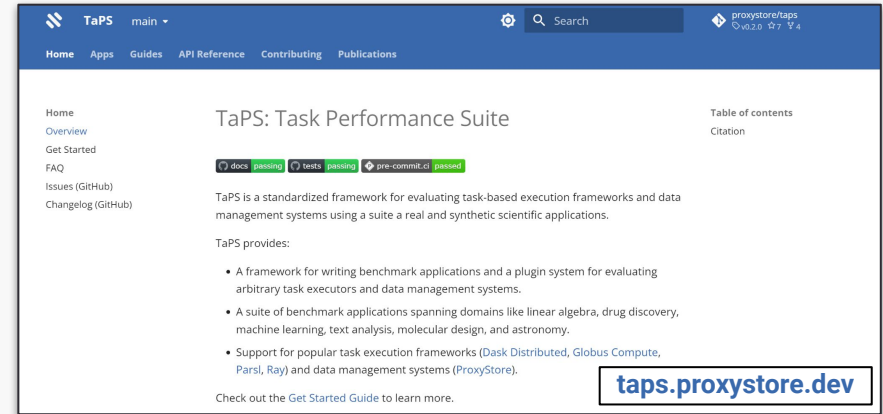
[†]Data Science and Learning Division, Argonne National Laboratory, Lemont, IL, USA

Abstract—Task-based execution frameworks, such as parallel programming libraries, computational workflow systems, and function-as-a-service platforms, enable the composition of distinct tasks into a single, unified application designed to achieve a computational goal. Task-based execution frameworks abstract the parallel execution of an application's tasks on arbitrary hardware. Research into these task executors has accelerated as computational sciences increasingly need to take advantage of parallel compute and/or heterogeneous hardware. However, the lack of evaluation standards makes it challenging to compare and contrast novel systems against existing implementations

uncover new areas for improvement. Benchmarks facilitate meaningful comparisons between competing approaches—a valuable aspect for researchers, reviewers, and readers alike.

Access to open source benchmarks democratizes research, and many fields have found great success through the creation of standards. LINPACK [4], for example, is used to evaluate the floating point performance of hardware systems. The Transaction Processing Performance Council (TPC) [5] provides a variety of standard benchmarks for database sys-

Try out TaPS



The screenshot shows the TaPS website interface. At the top, there's a navigation bar with 'TaPS main' and a search bar. Below that, a 'Home' section features a 'TaPS: Task Performance Suite' title and a 'Table of contents Citation' link. A progress bar shows the status of various benchmarks: 'docs' (passing), 'tests' (passing), and 'pre-commit.ci' (passed). The main content area describes TaPS as a standardized framework for evaluating task-based execution frameworks and data management systems. It lists several features: a framework for writing benchmark applications, a suite of benchmark applications in domains like linear algebra and drug discovery, and support for popular task execution frameworks like Dask and Globus Compute. A 'taps.proxystore.dev' badge is visible in the bottom right corner.

Good stuff to read in here 🙌

- more evaluation
- app descriptions
- technical details
- and more!

Want to collaborate? Reach out if you have...

- an app that could be a benchmark
- a new execution framework
- a data management system
- and more!



A Performance Evaluation Suite for Task-based Execution Frameworks



J. Gregory Pauloski



Valerie Hayot-Sasson



Maxime Gonthier



Nathaniel Hudson



Haochen Pan



Sicheng Zhou



Ian Foster



Kyle Chard

Questions?

Contact:

jgpauloski@uchicago.edu

github.com/proxystore/taps/issues

Reference:

<https://github.com/proxystore/taps>

<https://taps.proxystore.dev>

Acknowledgements:

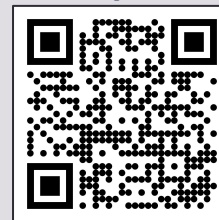
- Argonne National Laboratory under U.S. Department of Energy Contract DE-AC02-06CH1135
- National Science Foundation under Grant 2004894 and Grant 2209919
- Chameleon Cloud testbed supported by the National Science Foundation
- Cooperative Computing Lab at the University of Notre Dame

GitHub



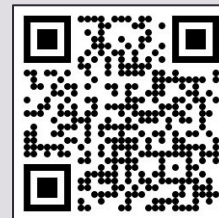
github.com/proxystore/taps

Preprint



arxiv.org/abs/2408.07236

Slides



regpauloski.com/#presentations