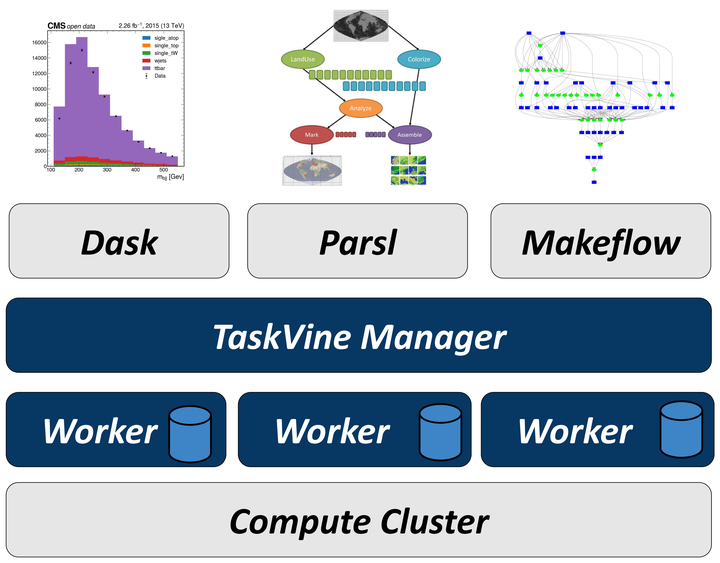
CSSI Element: ~~DataSwarm~~: **TaskVine:** A User-Level Framework for Data Intensive Scientific Applications

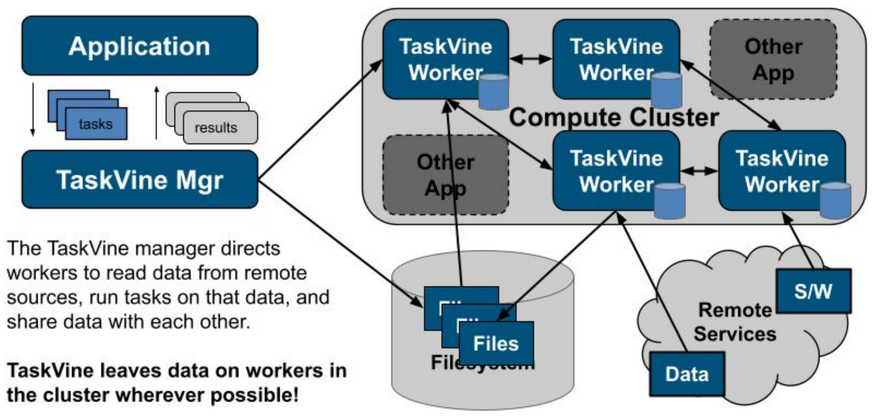
PI: Douglas Thain, University of Notre Dame, Award #: 1931348

TaskVine

http://cctools.readthedocs.io

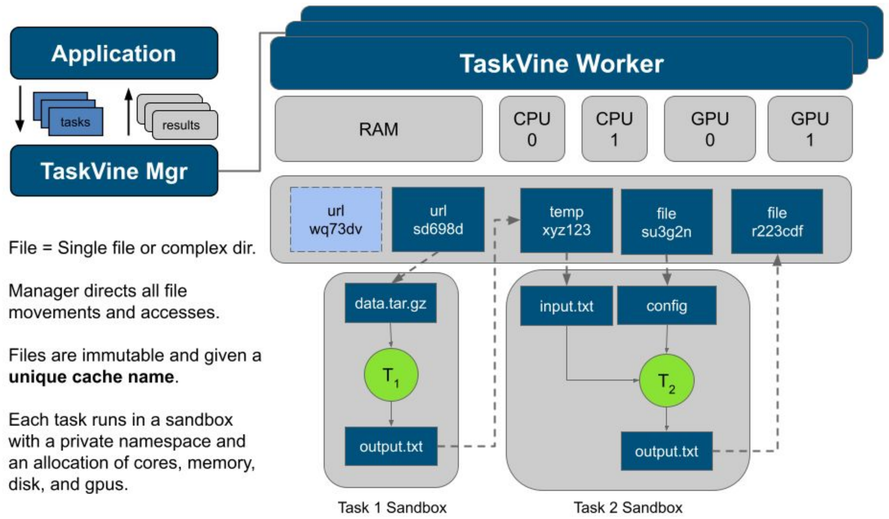
Many scientific applications are expressed as high-throughput **workflows** that consist of large graphs of data assets and tasks to be executed on large parallel and distributed systems. A challenge in executing these workflows is **managing data**: both datasets and software must be efficiently distributed to cluster nodes; intermediate data must be conveyed between tasks; output data must be delivered to its destination. Scaling problems result when these actions are performed in an uncoordinated manner on a shared filesystem. **TaskVine** is a system for exploiting the aggregate local **storage and network capacity** of a large cluster. TaskVine tracks the lifetime of data in a workflow – from archival sources to final outputs-- making use of local storage to distribute, and re-use data wherever possible.





The TaskVine manager directs workers to read data from remote sources, run tasks on that data, and share data with each other.

TaskVine leaves data on workers in the cluster wherever possible!



File = Single file or complex dir.

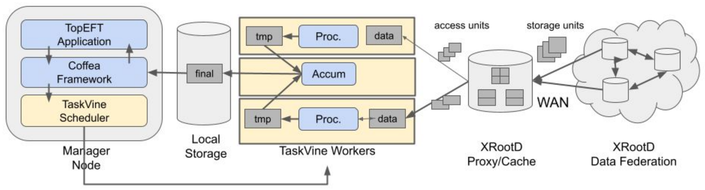
Manager directs all file movements and accesses.

Files are immutable and given a **unique cache name**.

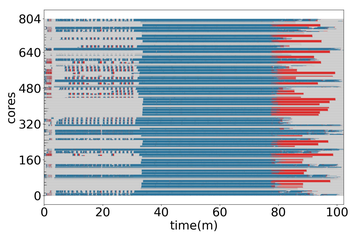
Each task runs in a sandbox with a private namespace and an allocation of cores, memory, disk, and gpus.

```
# Example of key TaskVine operaitons
import taskvine as vine
m = vine.Manager(9123)

# File objects are first class citizens.
file   = m.declareFile("mydata.txt")
buffer = m.declareBuffer("Some literal data")
url    = m.declareURL("https://nd.edu/data.tar.gz")
temp   = m.declareTemp();

# Peform a standard transformation on a file
data     = m.declareUntar( url )
software = m.declarePoncho( package )

# Submit a standard executable task.
task = vine.Task("mysim.exe -p 50 input.data -o
output.data")

t.add_input(url,"input.data")
t.add_output(temp,"output.data")
t.set_cores(4)
t.set_memory(2048)
t.set_disk(100)
taskid = m.submit(t)

# Submit a Python function execution
t = vine.PythonTask(
                simulate_func,molecule,parameters)
taskid = m.submit(t)

# Wait for any task to complete.
t = m.wait()
print( t.output )
```

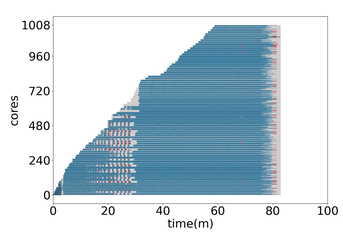
TopEFT Physics Data Analysis Application



Old: Work Queue

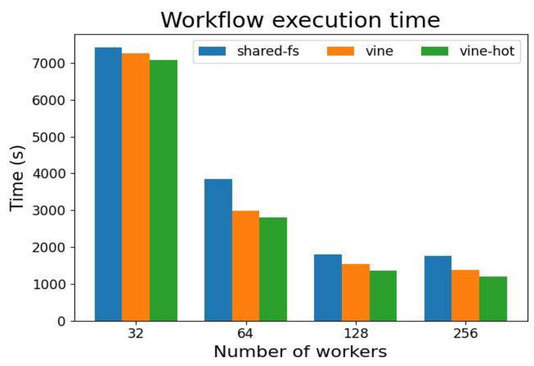


New: TaskVine



TaskVine Storage vs Shared Filesystem

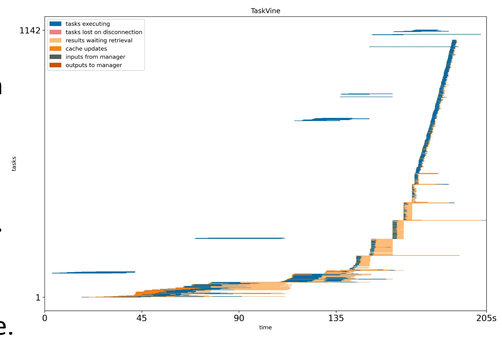
Effect of TaskVine's data management and distribution method compared to relying on a shared filesystem for data staging. Workflow: Training 2048 NN configurations, each 2 cores/2GB, with 8 tasks per worker on an HTCondor pool using either Panasas or TaskVine for data mgmt. Software deps are 4.4GB of Python libraries per node!



Multi-Modal Serverless Workflows

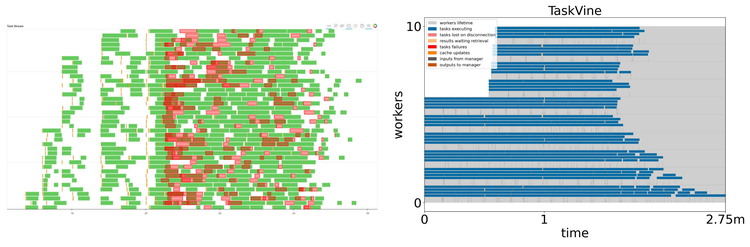
Simply converting "import tensorflow" into the preamble of a Library task saves **1.2GB** of Python libraries, **30K** metadata system calls, and **5-10s** latency per FunctionCall. We can mix standard Tasks, Libraries, and FunctionCalls into a multi-modal workflow:

100x Tasks build model from MNIST data. For each model, deploy serverless LibraryTask for inference. Submit 10x FunctionCalls to invoke each LibraryTask. Application gradually accelerates as Libraries become available to invoke.



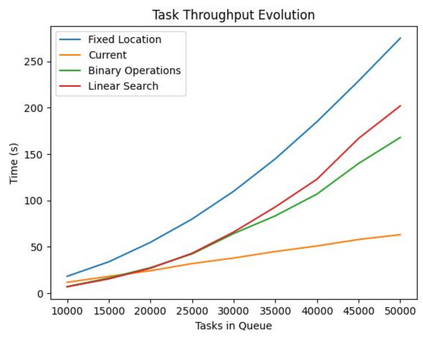
Dask.Distributed vs. Dask.TaskVine

For some workflows, TaskVine significantly outperforms Dask native distributed workers. The following workflow performs a series of matrix multiplication tasks forming a tree of task executions. 256 initial multiplication tasks feed into new tasks and son on, generating a final matrix. Each matrix is 1.2MB which is transferred between workers when needed.



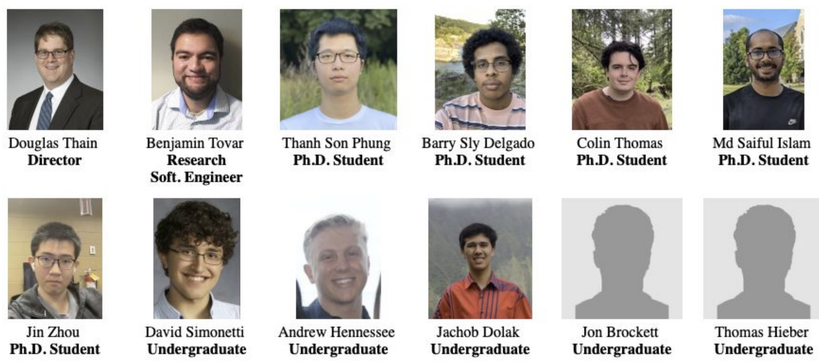
TaskVine Scheduler Scaling

The TaskVine scheduler must consider a number of aspects while making task placement and data movement decisions without compromising high-throughput task dispatch capabilities, especially for serverless applications. As the waiting queue grows, scheduling complexity can easily become an exponential function. Through several techniques we have improved to linear performance, and made greater scale possible.



Barry Sly-Delgado, Thanh Son Phung, Colin Thomas, David Simonetti, Andrew Hennesse, Ben Tovar, and Douglas Thain, **"TaskVine: Managing In-Cluster Storage for High Throughput Data Intensive Workflows",** WORKS Workshop at Scupercomputing, November 2023.

Cooperative Computing Lab at the University of Notre Dame



Douglas Thain, **Director**
Benjamin Tovar, **Research Soft. Engineer**
Thanh Son Phung, **Ph.D. Student**
Barry Sly Delgado, **Ph.D. Student**
Colin Thomas, **Ph.D. Student**
Md Saiful Islam, **Ph.D. Student**
Jin Zhou, **Ph.D. Student**
David Simonetti, **Undergraduate**
Andrew Hennessee, **Undergraduate**
Jachob Dolak, **Undergraduate**
Jon Brockett, **Undergraduate**
Thomas Hieber, **Undergraduate**