Scaling Up Throughput-oriented SLM Inference Applications on Opportunistic GPU Clusters with Pervasive Context Management

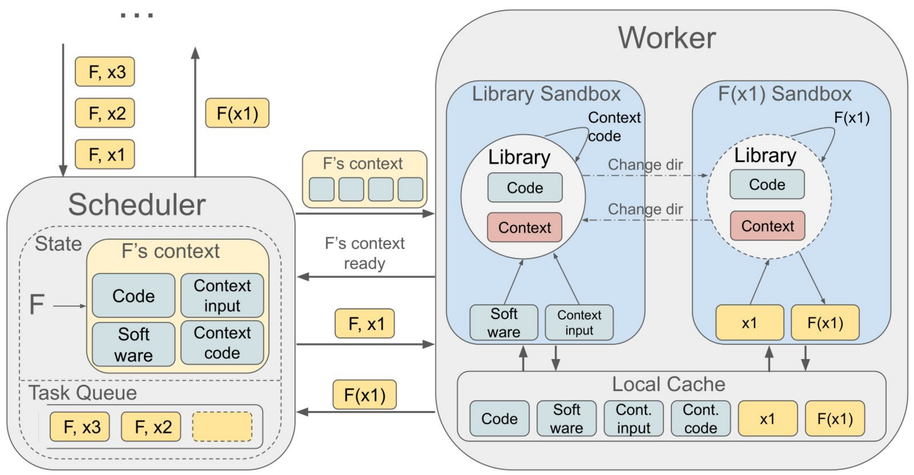
Thanh Son Phung
Cooperative Computing Lab
University of Notre Dame
tphung@nd.edu

Abstract

Executing large-scale, throughput-oriented Small Language Model (SLM) inference workloads on conventional shared clusters leads to **long job queues** due to inefficient static resource allocation. While opportunistic GPU clusters offer a vast pool of compute, their **transient** nature makes them **impractical**, as the **overhead** of repeatedly initializing an inference job's state with billions of parameters after preemption is prohibitively **high**.

We introduce Pervasive Context Management, a technique that makes SLM inference on opportunistic resources efficient. It works by **decoupling** the computational **context**, such as model weights and software dependencies, **from** the actual **inference execution**. This enables the **efficient reuse** and **rapid context transfer** between inferences across intermittently available GPUs, **virtually eliminating initialization costs**. Our evaluation shows this approach accelerates the end-to-end completion of large-scale SLM workloads by up to **3.6x** compared to traditional distributed executions, effectively scaling inference throughput on opportunistic resources.

Solution: Reusing Context with *Pervasive Context Management*



*Subsequent inferences of F reuse the same context in a remote worker's GPU, memory, and disk as managed by the **Library** process*

Implementation: Code Sample With *Pervasive Context Management*

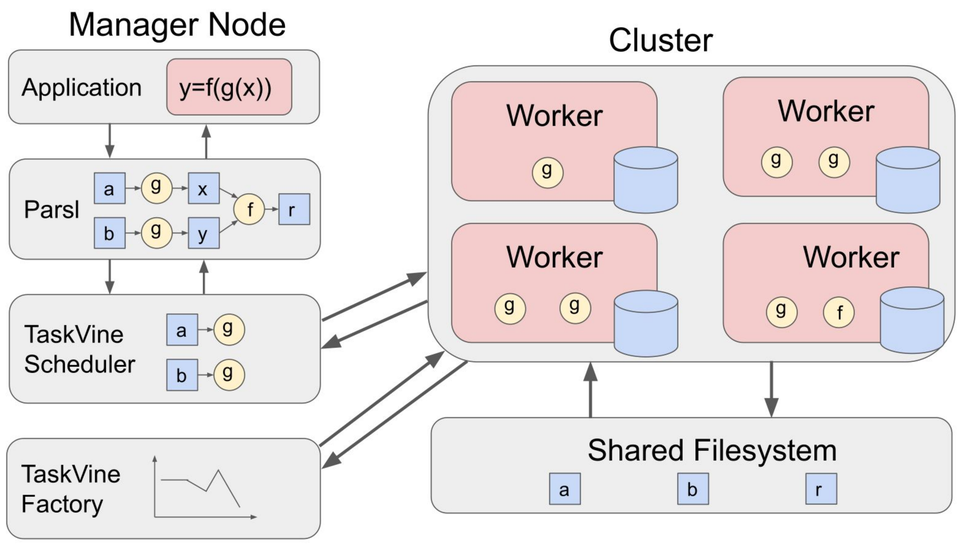
```python
from parsl import python_app
def load_model(model_path):
    ...
    model = AutoModel.from_pretrained(model_path).to('gpu')
    return {'model': model}

@python_app
def infer_model(inputs, parsl_spec):
    from parsl import load_variable_from_serverless
    model = load_variable_from_serverless('model')
    outputs = [model.generate(input) for input in inputs]
    return outputs

model_path = ...
parsl_spec = {'context': [load_model, [model_path], {}]}
inputs = ...
results = infer_model(inputs, parsl_spec).result()
```

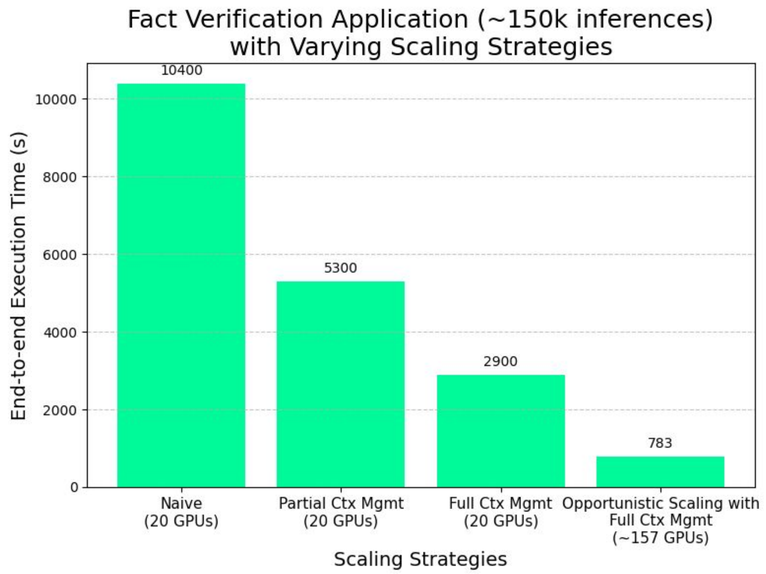
*This sample shows how the inference context - a model state retained in a remote GPU - is **decoupled** from an inference execution, allowing its efficient reuse for subsequent inferences*

Background: Parsl-TaskVine Parallel Framework



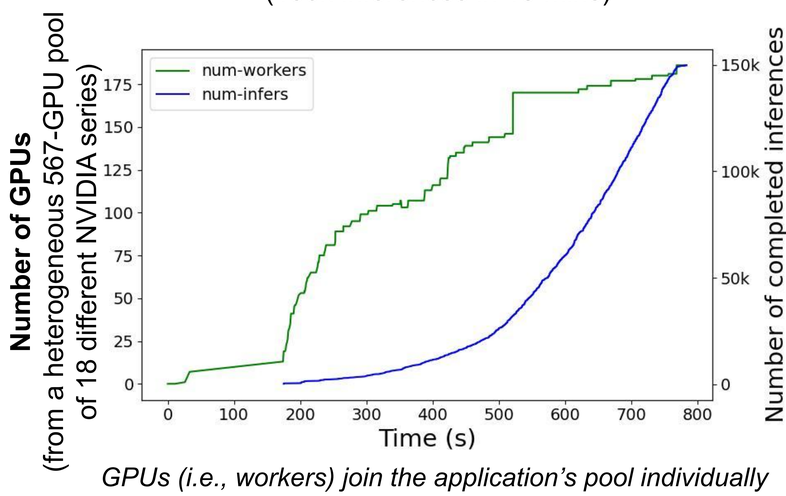
Functions generated from the application are passed to Parsl, which creates a DAG of functions and converts them into tasks for TaskVine Scheduler. Tasks are then scheduled to TaskVine Workers in the cluster, and TaskVine Factory moderates the number of workers as pre-specified.

Results



With a baseline of 20 GPUs, more context management results in much faster execution time. A free run that acquires opportunistic resources aggressively (i.e., Opportunistic Scaling) reduces the execution time of the application to 13 minutes.

Opportunistic Scaling of a Fact Verification Application with Aggressive Opportunistic Node Acquisition (150k inferences in 13 mins)



GPUs (i.e., workers) join the application's pool individually as soon as they are available, pushing the execution time of 150k inferences to 13 minutes

CCTools
UNIVERSITY OF NOTRE DAME