OrangeGrid Cloud Workers

Bursting an HTCondor pool into external capacity without changing the researcher experience.

Same workflow. Same submit files. Same storage model.

Just more capacity when we need it.

Peter Pizzimenti | Research Computing | Syracuse University

THROUGHPUT COMPUTING WEEK 2026

Today's story

Why, how, what we found, and where it goes.

1 Why

- A focused tool
- Why we built it

2 How

- Architecture overview
- Deployment
- Compute layer
- Storage layer

3 Results

- Performance
- Capacity planning
- Capacity from anywhere

4 Future

- What's next
- Key takeaways

A suite of tools with a focus

Different organizations have different constraints. This is the path that worked for us.

What this is

- Adds external capacity to an existing HTCondor pool
- Transparent to researchers
- Uses existing NFS home directories
- Built around our environment and constraints

What this is not

- A replacement for HTCondor Annex / Glidein
- A replacement for HTCondor file transfer
- A replacement for CVMFS / Pelican
- A universal solution

Why we built it

The situation

- Research demand is bursty
- Buying hardware for peak is expensive
- Researchers depend on NFS home directories
- We are all creatures of habit

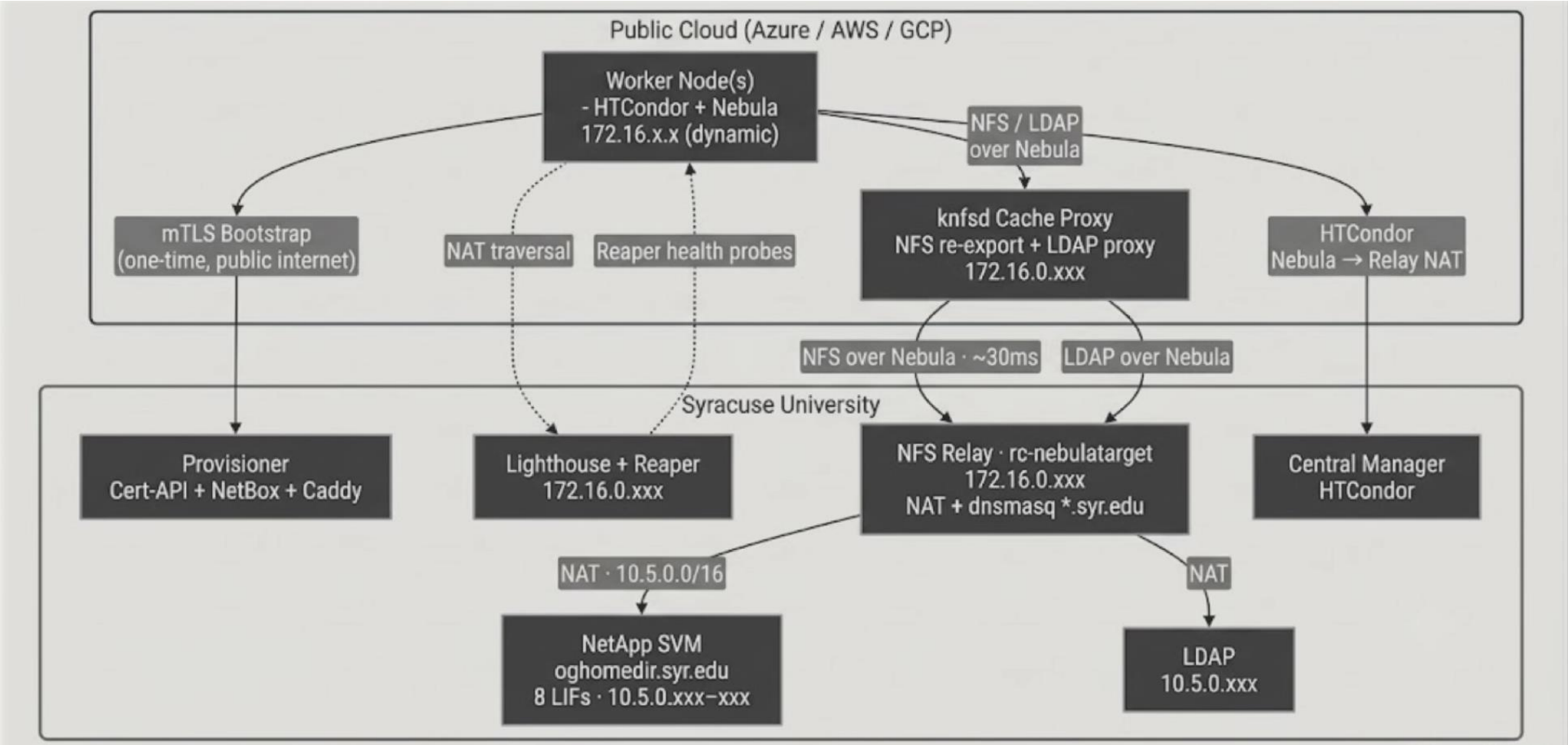
Requirements

- User changes nothing
- Data follows the workload
- Compute can come from anywhere
- Endpoint: Linux, Docker, Internet

The goal is adding JIT capacity without bifurcating the cluster.

Architecture overview

The home directory path, end to end. Cloud on top, campus below.



Cloud worker deployment

One phone call home.

Bundle contents

- Nebula certificate
- HTCondor software / creds
- Network config · NetBox
- Identity · LDAP

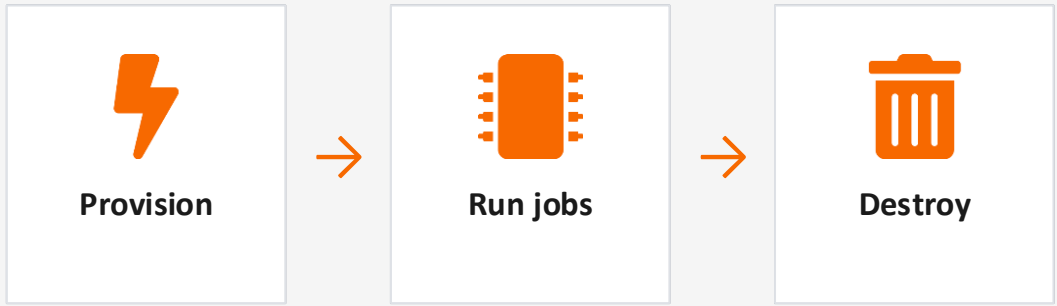
Compute layer

Disposable compute. It boots, joins the pool, runs jobs, and disappears.

Components

- Execute node with on-prem parity
- Nebula overlay client
- NFSv4.2 client / AutoFS
- LDAP
- Health monitoring

Lifecycle



One Docker Compose stack, on a node without special configuration.

Storage layer

The hard part. The home directory simply exists.

Design goals

- JIT staging available to all cloud workers
- No campus storage config changes
- No workflow changes

The cache fills itself when data is touched.

JIT staging. No user intervention.

Performance characteristics

What worked well and what didn't.

What it tests	On-prem min	Cloud avg min	Cloud vs on-prem	Why
One 20-core Blender render job	6.48	6.64	1.0x	Same runtime
Frame-farm render throughput	13.48	18.85	1.4x	Scheduling overhead
3-minute CPU throughput	3.08	3.23	1.0x	Same runtime
3-minute Python startup + compute	3.10	4.62	1.5x	Startup + metadata
3-minute R startup + compute	3.10	3.98	1.3x	Metadata RPC
20-core node-isolated HPCC benchmark	2.04	1.87	0.92x	Node-local CPU, memory, MPI; little NFS impact
1 GiB NFS write / read / hash	3.13	1.14	0.36x	Local read cache / Written to dirty pages
2,000-file metadata workflow	0.27	9.34	35.0x	WAN metadata latency

- Compute-heavy workloads land at 1.0x to 1.5x
- Existing researcher workflows run unchanged
- Metadata-intensive workflows are the primary limitation

Capacity planning MCP

Different questions, one live pricing source.

"I have $500. What's the most capacity I can rent for the next 24 hours?"

AMD64 CPU CAPACITY · SPOT

Provider	Instance type	vCPUs	24h cost
Azure	HB120rs_v3	3,720	$494.98
AWS	c5.12xlarge	2,976	$493.61
GCP	n2d-highcpu-224	3,808	$498.45

Best: GCP, 3,808 vCPUs for ~$498

"How long should we rent cloud GPUs to clear today's HTCondor's GPU queue?"

GPU QUEUE · SPOT

Item	Value
Current queue	547 waiting GPU jobs
Job shape	1 GPU, 16 vCPU, 16 GiB RAM
Runtime basis	Successful GPU job history from users: <NetIDs>
P99 runtime	~22 hours
Recommended rental	24 hours
Conservative rental	30 hours
Best 24h spot estimate	Azure A10 $3.9k, AWS L4 $3.9k

Once every job can start, rental time is driven by tail runtime.

Capacity from anywhere

Cloud-agnostic by design. The worker only needs to know where home is.

RUNNING TODAY

Azure — *where it was built and tested*

POSSIBLE TOMORROW

AWS
Public cloud

GCP
Public cloud

Neoclouds
GPU specialists

Another university's cloud
Peer capacity

Private infrastructure
On-prem / lab

What’s next

Where this goes from here.

1 Open to all OrangeGrid users

Graduate from beta so any SU HTCondor researcher can consume.

2 Automate cloud consumption

Use live cross-cloud pricing to drive spot bids to a budget.

Historical job and performance data to better route work to either keep on-prem or send to the cloud

3 Burst GPUs

Extend from CPU to GPU execute nodes for accelerated work.

4 Reach more clouds

Move past Azure only and prove out AWS, GCP, and a private-cloud targets.

Key takeaways

- Researchers do not change workflows
- Capacity can come from anywhere
- Compute-heavy HTC workloads perform well
- Metadata-heavy workflows remain the primary limitation
- Another HTCondor deployment option, not a replacement

docs.syr.edu/ResearchComputing/cloudbursting

The hard part was never getting the compute.

It was making that remote compute fit our researcher's familiar workflow.

Questions?