HYPHA

Resilient Heterogeneous Distributed Inference

Experimental validation of throughput scaling, forced rank-loss detection, and automated pipeline recovery using GPT-OSS 20B across an RTX A6000 and RTX 3060

Technical White Paper - Experimental Results
Version 0.1 | September 10, 2026

Key result
A repeated, matched 12/12 pipeline split averaged 36.820 tokens/s versus 21.509 tokens/s for the A6000-only control, a 71.2% throughput increase. A separate forced-failure experiment then killed the RTX 3060 inference rank with SIGKILL; Hypha detected and contained the failure, reclaimed resources, rebuilt the 12/12 pipeline, retried inference, and completed a 1,024-token response without operator intervention after the failure was detected.

Hypha Compute

Abstract

This paper reports a controlled experimental demonstration of heterogeneous distributed large-language-model inference and recovery under forced component failure. The system under test used openai/gpt-oss-20b with 24 hidden transformer layers, an NVIDIA RTX A6000 as pipeline rank 0, and an NVIDIA RTX 3060 as pipeline rank 1. vLLM 0.29.0 executed pipeline-parallel inference, while Hypha provided the registry, node selection, job leasing, operation placement, lifecycle cleanup, and recovery orchestration used to coordinate the distributed run.

A partition sweep showed increasing throughput as work was shifted from the A6000 to the RTX 3060. A subsequent interleaved nine-run validation compared 13/11, 12/12, and a matched A6000-only control. The 12/12 split averaged 36.820 tokens/s, compared with 21.509 tokens/s for the control, a 71.2% increase. The validated split consumed 187.8 W mean combined GPU power and 1.4511 Wh per 1,024 completion tokens, compared with 105.0 W and 1.3889 Wh for the control.

A forced-failure test then sent SIGKILL to the RTX 3060 inference rank during active generation. The peer process reported an NCCL remote-process failure, the in-flight generation failed rather than returning corrupted output, active leases cleared, cleanup found no stale inference process or held lock, and the registry remained healthy. In the next experiment, the same failure was followed by automated Hypha-orchestrated cleanup, pipeline relaunch, and request retry. The rebuilt pipeline completed 1,024 tokens at 36.567 tokens/s. Measured failure-to-recovered-answer time was 244.6 seconds, with 136.7 seconds attributed to the automated recovery workflow after failure handling began.

Claim boundary

The experiments prove automated recovery orchestration using Hypha operations after a catastrophic inference-rank failure. They do not yet prove continuation of the same in-flight generation, recovery from total physical node loss, or a registry-native self-healing policy. The next engineering step is to move the demonstrated recovery state machine into Hypha itself.

1. Problem and Motivation

Distributed AI at the edge is constrained by different factors than centralized datacenter inference. Local systems may contain unequal GPUs, mixed generations of hardware, intermittent networks, limited power, and no guarantee that every node remains available. A useful local compute fabric must therefore do more than divide a model across devices. It must discover resources, place work on compatible hardware, detect and contain failures, release stale allocations, and restore useful service without depending on a remote cloud control plane.

The central question in this experiment was whether a heterogeneous pair of commodity GPUs could be treated as one coordinated inference fabric and, after a deliberately induced rank failure, whether the orchestration layer could return the service to a known-good state and rebuild it automatically.

2. System Under Test

Component	Role	Configuration
Hypha registry	Control plane	http://10.11.228.211:8080; sole active registry during the experiment
RTX A6000	Pipeline rank 0	48 GB class GPU; node node-4678f275-77e3-44d2-b761-98c22d7e047a; host 10.11.211.153
RTX 3060	Pipeline rank 1	12 GB class GPU; node node-7c4db209-4bf6-48d4-b4aa-0157f7b9d303; host 10.11.241.52
Model	Workload	openai/gpt-oss-20b; 24 hidden transformer layers
Runtime	Inference engine	vLLM 0.29.0; PP=2, TP=1; eager execution

Division of responsibility

- vLLM executes the model and pipeline-parallel data path.
- Hypha coordinates node identities, job placement, leases, operation installation, lifecycle cleanup, and the recovery workflow used in these experiments.
- NCCL provides the GPU-to-GPU distributed communication path used by the vLLM pipeline.
- The experiment intentionally used unequal GPUs on different hosts to test heterogeneous placement rather than a symmetric multi-GPU workstation.

3. Experimental Methodology

3.1 Matched inference configuration

- Pipeline parallel size: 2
- Tensor parallel size: 1
- VLLM_PP_LAYER_PARTITION changed by test partition only
- --enforce-eager
- --gpu-memory-utilization 0.80
- --max-model-len 4096
- --max-num-seqs 1
- VLLM_USE_FLASHINFER_SAMPLER=0
- Same benchmark prompt

- 1,024 completion tokens for the throughput validation

The partition sweep changed one model-placement variable: the number of hidden layers assigned to the A6000 versus the RTX 3060. All tested partitions summed to 24 hidden layers.

3.2 Validation design

To reduce order and thermal bias, the final validation interleaved three configurations across three cycles: 13/11, 12/12, and A6000-only 24/0. Each configuration therefore produced three matched samples. The experiment also collected GPU power, energy per 1,024 completion tokens, and tokens per joule.

3.3 Failure injection

The first resilience experiment started the validated 12/12 pipeline and began a long generation. A dedicated failure-injection endpoint on the RTX 3060 worker armed a delayed fault and then sent SIGKILL to the vLLM process group. The Hypha node service itself was not killed. This distinction matters: the test represents catastrophic loss of an inference rank/process, not complete loss of the physical host or Hypha node daemon.

4. Heterogeneous Throughput Scaling

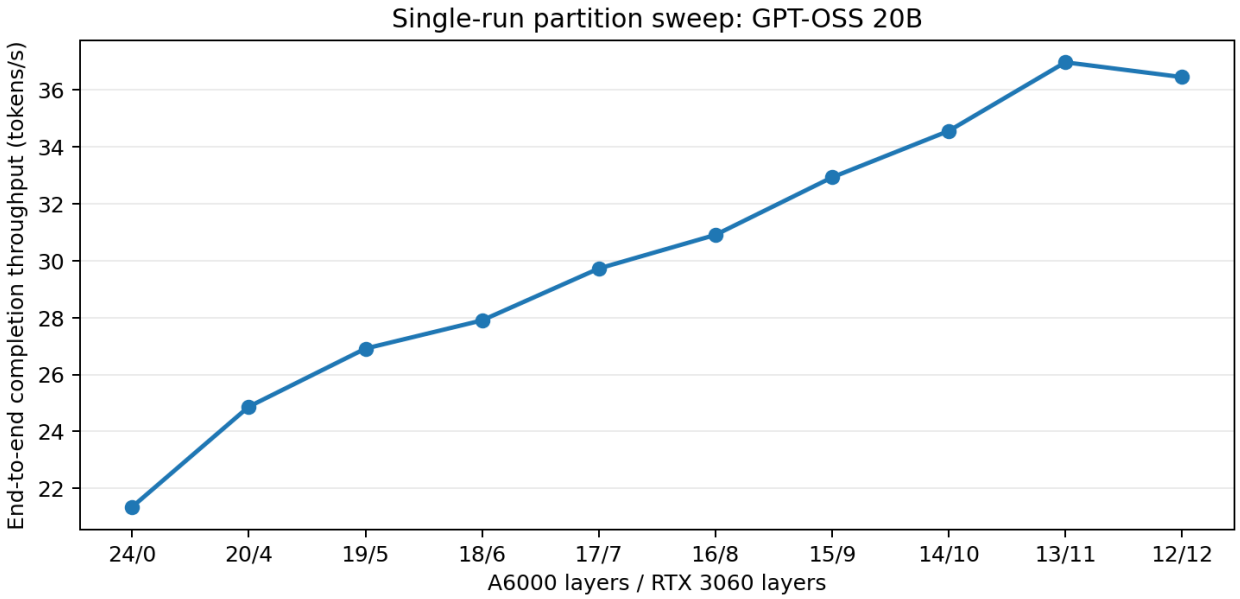

Figure 1. Single-run partition sweep. The values are exploratory measurements used to locate the high-performing boundary before repeated validation.

Partition	TPS	Status
24/0	21.35	Matched control
20/4	24.87	Sweep sample
19/5	26.91	Sweep sample

Partition	TPS	Status
18/6	27.90	Sweep sample
17/7	29.72	Sweep sample
16/8	30.90	Sweep sample
15/9	32.92	Sweep sample
14/10	34.55	Sweep sample
13/11	36.96	Validated candidate
12/12	36.44	Validated candidate

5. Repeated Validation Results

Config	n	Mean TPS	SD	Range	Mean W	Wh/1024	Tokens/J
24/0 control	3	21.509	0.343	21.056 - 21.887	105.0	1.3889	0.2050
13/11	3	35.985	0.207	35.693 - 36.152	185.6	1.4673	0.1939
12/12	3	36.820	0.235	36.493 - 37.039	187.8	1.4511	0.1960

Validated throughput result

The 12/12 split averaged 36.820 tokens/s versus 21.509 tokens/s for the matched A6000-only control, a 71.2% increase. The three 12/12 samples ranged from 36.493 to 37.039 tokens/s. The three 13/11 samples ranged from 35.693 to 36.152 tokens/s; the ranges did not overlap in this nine-run validation.

The power result is equally important. The 12/12 split used substantially more instantaneous GPU power (187.8 W combined mean versus 105.0 W for the A6000-only control), but the higher throughput shortened generation time enough that measured energy per 1,024 completion tokens rose only from 1.3889 Wh to 1.4511 Wh, approximately 4.5%. In this experiment, a 71.2% throughput increase therefore required only a modest increase in measured GPU energy per fixed-size completion.

6. Forced Rank-Loss Detection

The first resilience experiment deliberately destroyed pipeline rank 1 during active generation. The RTX 3060 worker reported exit status -9 and confirmed that the failure injector had fired. Rank 0 on the A6000 observed an NCCL remote-process failure and the connection closed by the remote peer. The generation did not complete normally.

Observation	Result
RTX 3060 forced failure	SIGKILL fired; worker exit_status = -9
Peer detection	NCCL remote process/network error; remote peer closed connection
Generation integrity	generation_completed = false
Head active lease	False after terminal state
Worker active lease	False after terminal state
Lease audit	lease_audit_ok = true for both jobs
Post-failure cleanup	No stale vLLM worker; no held inference lock
Registry	healthz = ok; queue = 0; agents = 15; active_agents = 15

What this proves

A catastrophic inference-rank loss was detected and contained without a false-success response, leaked active lease, stale worker, or registry failure. This is failure detection and state hygiene, not yet transparent continuation of the original request.

7. Automated Pipeline Recovery

The second resilience experiment repeated the same forced rank loss and then, with no operator intervention after failure detection, invoked Hypha operations to reclaim resources, create a replacement RTX 3060 worker job, create a replacement A6000 head job, rebuild the 12/12 pipeline, and retry inference. The replacement request completed exactly 1,024 completion tokens.

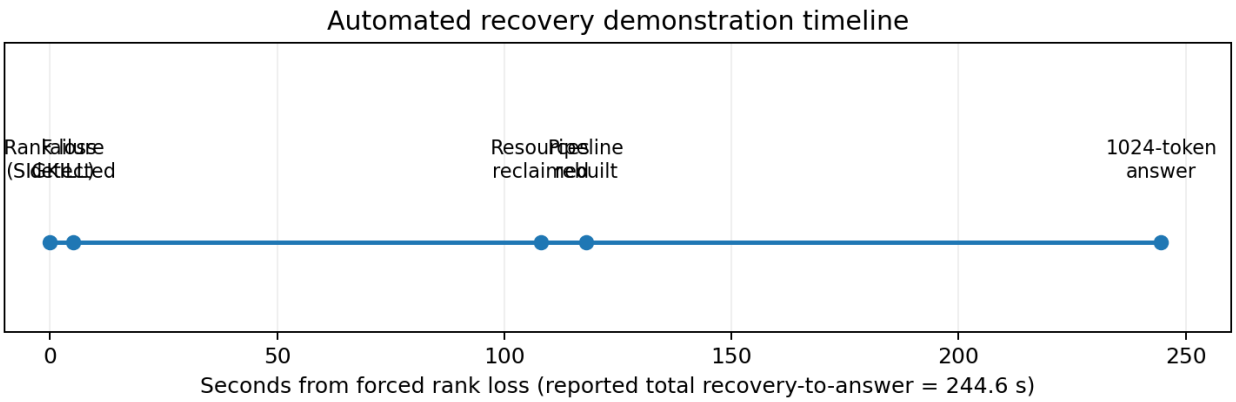

Figure 2. Recovery demonstration. Intermediate markers are conceptual; the reported measured endpoints are 244.6 seconds failure-to-recovered-answer and 136.7 seconds for the automated recovery workflow.

Recovery metric	Observed value
Forced rank loss	True
Failed request rejected	True
Failed leases reclaimed	True
Pipeline rebuilt automatically	True
Retry completed	1,024 completion tokens
Recovered throughput	36.567 tokens/s
Failure-to-recovered-answer	244.6 s
Automated recovery workflow	136.7 s
Registry healthy after recovery	True

Recovery claim

This experiment demonstrates automated recovery orchestration after a catastrophic distributed inference-rank failure. The recovery logic was driven by an external coordinator script that used Hypha jobs and cleanup operations. The next step is to promote that same recovery behavior into a Hypha-native policy/state machine so the fabric itself owns the decision loop.

8. Why the Result Matters

8.1 Heterogeneous hardware becomes a schedulable fabric

The A6000 and RTX 3060 differ substantially in memory capacity and performance. Nevertheless, the validated 12/12 split shows that useful model execution can be distributed across them without requiring a homogeneous GPU server. The result supports a design direction in which model placement is treated as a scheduling problem over discovered resources rather than a static mapping fixed by hardware symmetry.

8.2 Capacity and throughput are both system-level properties

The experiment did more than use a second GPU as overflow storage. The second device performed enough useful computation to increase end-to-end throughput materially. This suggests that future Hypha scheduling can optimize across multiple objectives: fit, throughput, energy, resource availability, and resilience.

8.3 Failure behavior is more important than a benchmark peak

A local compute fabric deployed under intermittent connectivity or field conditions must assume that components fail. The forced-rank-loss experiments therefore represent a more meaningful systems test than another incremental throughput benchmark. The significant observation is not that the rank crashed; it is that the orchestration state remained coherent and that service could be rebuilt and verified automatically.

9. Limits and Non-Claims

- The experiment used one A6000 and one RTX 3060. It does not yet prove scaling behavior across three or more inference nodes.
- The failure injection killed the vLLM inference rank/process, not the entire physical machine, network interface, Hypha node daemon, or power supply.
- The same in-flight generation was not resumed from a checkpoint. The failed request was rejected and a new pipeline was built for a retry.
- The automatic recovery decision loop was implemented by an experiment coordinator using Hypha operations. It is not yet a built-in registry-native recovery policy.
- Three samples per validated configuration are sufficient to establish repeatability for this engineering demonstration, but not to characterize every source of long-run performance variance.
- GPU energy values are based on nvidia-smi power telemetry during the benchmark window and do not include whole-system CPU, memory, switch, storage, or PSU losses.
- No claim is made here that Hypha is the only platform capable of distributed inference. The demonstrated distinction is the combination of heterogeneous resource orchestration, lease-aware lifecycle control, failure containment, and automated rebuild behavior within the Hypha control model.

10. Checkpoint and Immediate Next Step

Current checkpoint
THROUGHPUT VALIDATION: PASSED
FORCED RANK-LOSS DETECTION: PROVEN
AUTOMATED PIPELINE RECOVERY: PROVEN
NEXT = MAKE_RECOVERY_POLICY_HYPHA_NATIVE

The next engineering task is not another throughput benchmark. It is to move the demonstrated recovery state machine into Hypha proper. The native policy should observe job/lease failure, identify an eligible replacement placement, reclaim stale resources, rebuild the required distributed ranks, retry according to a bounded policy, and emit auditable recovery events.

Native recovery acceptance criteria

- No external experiment coordinator is required after the workload is submitted.
- Failure detection is triggered from Hypha-owned state/events rather than a polling script that owns the recovery decision.
- Original failed leases become terminal and are not reused incorrectly.
- Replacement jobs are placed only on eligible healthy nodes with sufficient resources.
- Retry count and backoff are bounded and auditable.
- A recovered request completes successfully, or the recovery policy terminates with an explicit final failure state.
- Registry health and unrelated node workloads remain unaffected.
- Recovery timing is recorded as a first-class metric.

11. Reproducibility Record

The following experimental constants should be preserved when reproducing the reported throughput and recovery tests:

- Model: openai/gpt-oss-20b
- Hidden layers: 24
- vLLM: 0.29.0
- Pipeline parallel size: 2
- Tensor parallel size: 1
- Validated partition: 12/12
- A6000 node: node-4678f275-77e3-44d2-b761-98c22d7e047a
- RTX 3060 node: node-7c4db209-4bf6-48d4-b4aa-0157f7b9d303
- Registry: http://10.11.228.211:8080
- --enforce-eager
- --gpu-memory-utilization 0.80
- --max-model-len 4096
- --max-num-seqs 1
- VLLM_USE_FLASHINFER_SAMPLER=0

- Failure injection: SIGKILL to the RTX 3060 vLLM process group during active generation

The original validation and recovery logs should be retained alongside the code package. For external sharing, the experimental coordinator should be separated from production Hypha code and clearly labeled as a reproducibility harness until the recovery policy is implemented natively.

12. Conclusion

The experiments establish a concrete systems result: a heterogeneous two-host GPU pipeline coordinated through Hypha substantially increased GPT-OSS 20B throughput relative to a matched A6000-only control, survived a deliberately injected inference-rank failure without corrupting orchestration state, and was then rebuilt and retried automatically using Hypha operations. The recovered service returned to the same performance regime as the validated pipeline.

The most important next milestone is architectural rather than numerical. By moving the demonstrated recovery logic into a Hypha-native policy, the system can progress from an externally driven recovery demonstration to a fabric that owns failure detection, placement, cleanup, retry, and audit as part of its normal control plane. That is the next checkpoint.

Appendix A. Evidence Summary

Evidence item	Recorded result
Nine-run interleaved validation	12/12 mean 36.820 TPS; control mean 21.509 TPS; +71.2%
12/12 mean GPU power	187.8 W combined
12/12 mean energy	1.4511 Wh / 1,024 completion tokens
Control mean energy	1.3889 Wh / 1,024 completion tokens
Forced-failure worker status	SIGKILL fired; exit_status = -9
In-flight request	Rejected; generation_completed = false
Lease state after failure	No active head or worker lease; lease audits clean
Registry after failure	ok; queue 0; 15 active agents
Recovered inference	1,024 completion tokens at 36.567 TPS
Failure-to-recovered-answer	244.6 seconds

Automated recovery workflow	136.7 seconds
Final checkpoint	HYPHA_AUTOMATIC_FAILURE_RECOVERY=PROVEN

Source record

Results in this white paper are derived from the September 10, 2026 Hypha GPT-OSS partition-sweep, nine-run validation, forced-rank-loss, and automated-recovery console logs supplied with the experiment. No external benchmark claims are used.

HYPHA | Patent Pending

© 2026 TKT, Inc. All Rights Reserved.

HYPHA technology is the subject of pending patent applications. The contents of this white paper, including its text, illustrations, diagrams, and technical descriptions, are protected by applicable copyright laws. No portion may be reproduced, distributed, or transmitted without prior written permission from TKT, Inc.