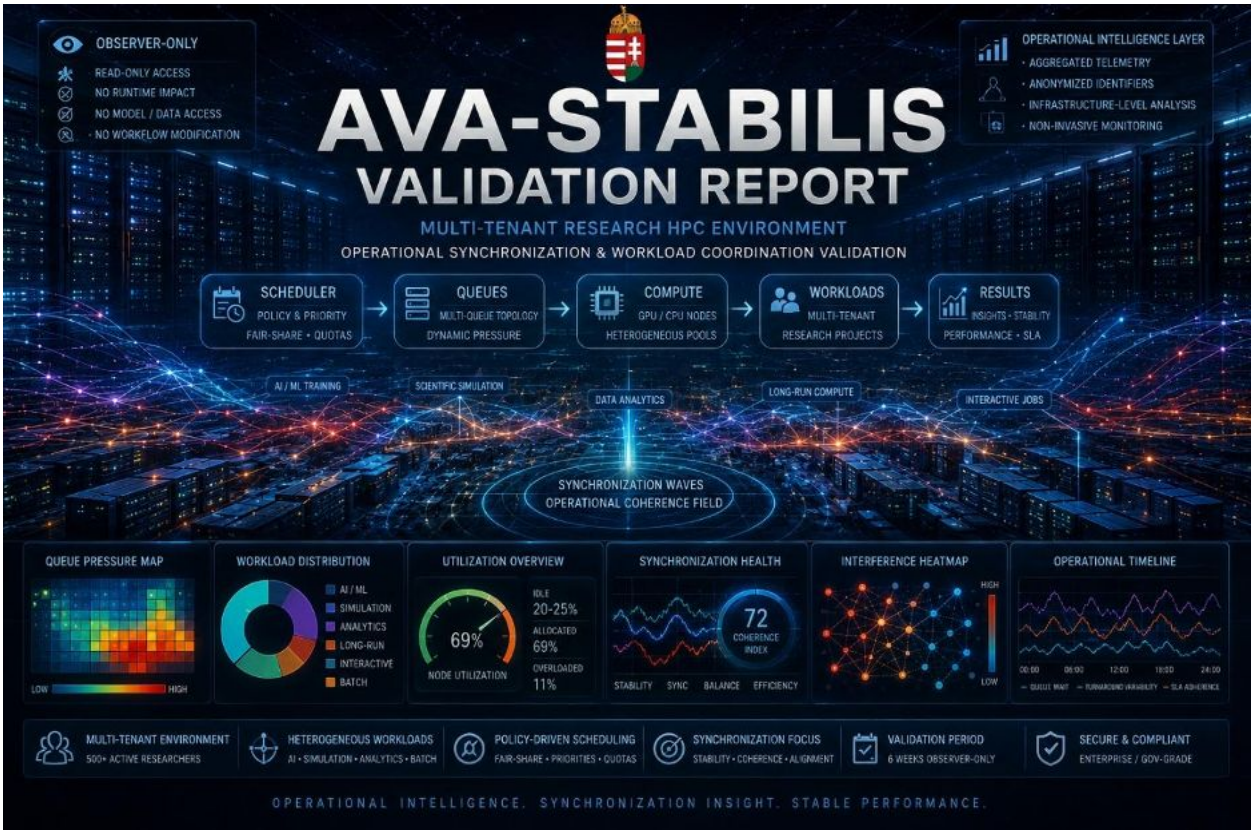
AVA-STABILIS

Operational Synchronization Modeling Report

MULTI-TENANT RESEARCH WORKLOAD ENVIRONMENT



This document is a partially modeled operational-analysis sample report. It is designed to demonstrate the AVA-Stabilis methodology and does not represent an audited customer deployment or a peer-reviewed scientific validation.

- **Pilot Type:** Operational Synchronization Validation
- **Environment:** Multi-tenant Academic / Research HPC
- **Version:** v1.0
- **Date:** [Insert Date]
- **Confidentiality Level:** Confidential / Internal Use Only
- **Operational Model:** Observer-Only / Read-Only Analysis
- **Partner Organization:** Anonymized

---

0. EXECUTIVE SUMMARY

Pilot Objective

The objective of the pilot was to investigate how operational instability emerges within a shared multi-tenant research computing environment characterized by:

- heterogeneous computational workloads,
- multiple concurrent research users,
- policy-driven scheduling mechanisms,
- and dynamically competing workload priorities.

The pilot specifically focused on identifying:

- operational instability patterns,
- scheduling-induced distortions,
- queue propagation dynamics,
- and cross-workload interference effects
  inside a live academic HPC environment.

The investigation was not intended to redesign the infrastructure itself,
but rather to analyze how operational synchronization dynamics influenced overall system behavior.

Observation Period

The validation was conducted during a structured 6-week observer-only analysis period:

- within a live production environment,
- under normal operational conditions,
- and without external runtime intervention or infrastructure modification.

The pilot followed a strictly:

- read-only,
- non-invasive,
- aggregated telemetry-based
  operational analysis methodology.

Key Operational Challenge

The investigated environment demonstrated formally fair resource allocation behavior from a scheduling-policy perspective.

However, operational analysis revealed:

- significant low-level operational noise,
- queue instability patterns,
- cross-workload interference propagation,
- fragmented scheduling behavior,
- and hidden capacity losses
  despite apparently acceptable infrastructure utilization metrics.

The pilot identified a recurring structural paradox:

- idle compute capacity
  coexisting simultaneously with
- persistent queue congestion
  and delayed workload execution.

Key Findings

The investigation identified several recurring operational patterns:

- fairness ≠ operational efficiency
- heterogeneous workloads generated cross-interference effects
- scheduler policy logic amplified instability propagation
- idle capacity and queue congestion appeared simultaneously
- workload fragmentation reduced global coordination efficiency
- system behavior was constrained primarily by policy synchronization dynamics rather than by raw infrastructure limitations

The pilot demonstrated that many observed inefficiencies originated from:

- scheduling conflicts,
- delayed balancing reactions,
- and workload synchronization mismatches
  rather than from insufficient compute resources.

Key Validated Results

The validation phase demonstrated measurable operational improvements through limited policy-level synchronization adjustments:

- reduced average queue wait times
- improved node utilization
- reduced workload interference propagation
- improved turnaround consistency
- reduced operational fragmentation
- improved SLA adherence
- lower hidden idle-capacity ratios

Importantly, these improvements were achieved:

- without infrastructure expansion,
- without additional compute resources,
- and without modifying the underlying HPC architecture.

Strategic Conclusion

“In multi-tenant research environments, operational performance is determined less by available compute capacity and more by synchronization quality between workloads, policies, and scheduling dynamics.”

1. SYSTEM UNDER INVESTIGATION

Environment Type

The investigated environment was a large-scale academic and research-oriented HPC infrastructure operating as a shared multi-tenant compute environment.

The system supported:

- concurrent research workloads,
- heterogeneous computational demands,
- and dynamically changing execution priorities
  across multiple scientific and engineering domains.

The infrastructure operated as a centrally coordinated shared compute platform designed to serve:

- research institutions,
- university departments,
- AI-related experimentation,
- simulation workloads,

- and data-intensive computational activities.

User Environment

The environment supported:

- more than 500 active researchers,
- multiple institutional departments,
- and parallel research projects with significantly different computational behavior profiles.

The investigated user population included:

- AI researchers,
- scientific computing teams,
- data analytics users,
- simulation-heavy research groups,
- and mixed interactive / batch-oriented workloads.

As a consequence, the system continuously experienced:

- heterogeneous demand waves,
- competing execution priorities,
- and highly variable runtime characteristics.

Workload Types

The analyzed environment contained multiple workload categories operating simultaneously within the same shared scheduling ecosystem.

Observed workload classes included:

- AI / ML training workloads,
- scientific simulations,
- large-scale data analytics,
- long-running compute-intensive jobs,
- interactive research workloads,
- and batch-processing operations.

These workload types demonstrated significantly different:

- runtime durations,
- memory consumption patterns,
- GPU utilization characteristics,
- queue sensitivity,
- and scheduling responsiveness requirements.

The coexistence of these heterogeneous workload categories generated persistent operational synchronization challenges throughout the environment.

Scheduler Environment

The investigated cluster utilized a:

- multi-queue,
- policy-driven scheduling architecture
  based on:
- fair-share allocation principles,
- quota-controlled execution,
- and segmented priority management.

The scheduler continuously attempted to:

- balance fairness,
- maintain utilization,
- distribute compute access,
- and enforce institutional policy constraints
  across competing workloads and user groups.

Operational analysis revealed that:
while fairness objectives were largely maintained from a policy perspective,
the resulting runtime behavior frequently produced:

- fragmented execution patterns,
- unstable queue topology,
- and synchronization conflicts between workload classes.

Operational Characteristics

The environment demonstrated several recurring operational characteristics typical of large-scale multi-tenant research systems:

- highly heterogeneous workload dynamics,
- burst-like resource demand patterns,
- nonlinear queue propagation behavior,
- competing scheduling objectives,
- and continuously shifting execution pressure across compute pools.

The system exhibited:

- localized overload states,
- delayed balancing reactions,
- transient idle-capacity regions,
- and recurring queue-wave propagation effects
  that collectively contributed to operational instability.

The observed behavior suggested that the environment functioned not merely as a static compute infrastructure,
but rather as a dynamically interacting operational field with continuously evolving synchronization conditions.

Access Model

The pilot followed a strictly observer-only operational model.

The investigation was conducted using:

- read-only telemetry,
- aggregated operational metrics,
- anonymized scheduling information,
- and non-invasive operational analysis techniques.

The pilot explicitly excluded:

- runtime intervention,
- infrastructure-level modification,
- workload manipulation,
- direct scheduler control,

- and access to model contents or research data.

No:

- training data,
- model weights,
- user-level research content,
- or confidential computational payloads were accessed during the investigation.

The analysis focused exclusively on:

- operational dynamics,
- scheduling behavior,
- synchronization patterns,
- and infrastructure-level execution topology.

---

2. BASELINE OPERATING STATE

Baseline Metrics

Metric	Baseline
Average queue wait	~7.8 h
Node utilization	~69%
Idle node ratio	~20–25%
Turnaround variability	High
SLA adherence	~76%
Cross-workload interference	Frequent
Priority inversion events	Frequent

Baseline Interpretation

From a policy perspective, the investigated environment demonstrated formally fair operational behavior through:

- fair-share allocation,

- quota balancing,
- and policy-driven scheduling logic.

However, operational analysis revealed that the system functioned in a partially fragmented and internally conflicting state.

The environment exhibited:

- oscillatory scheduling behavior,
- unstable queue propagation dynamics,
- and recurring synchronization mismatches between workload classes.

Several recurring structural patterns emerged during the baseline observation phase:

- idle-capacity and overload states appeared simultaneously,
- downstream queue-wave propagation effects amplified local scheduling disturbances,
- and scheduler-induced instability continuously reshaped execution topology.

The system therefore operated in a condition where:

- local scheduling decisions frequently conflicted with global operational efficiency,
- resulting in hidden capacity losses and reduced overall synchronization quality.

Operational evidence suggested that a substantial portion of the observed inefficiency originated not from insufficient compute capacity,
but from:

- policy fragmentation,
- delayed balancing reactions,
- workload interference,
- and unstable coordination dynamics between heterogeneous workloads.

3. IDENTIFIED OPERATIONAL PATTERNS

3.1 Fairness–Efficiency Conflict

Observed Phenomenon

The implemented fair-share scheduling model provided formally balanced resource access across users and research groups.

However, the scheduling logic primarily optimized for:

- policy-level fairness,
  while only partially accounting for:
- workload-specific execution characteristics,
- runtime behavior,
- locality constraints,
- and synchronization requirements between heterogeneous workloads.

As a consequence, workloads with fundamentally different operational profiles were treated similarly from a scheduling perspective despite significantly different execution dynamics.

Operational Chain

fair-share balancing
→ fragmented resource allocation
→ reduced workload locality
→ lower execution efficiency
→ hidden capacity loss

Consequence

The resulting operational effects included:

- reduced overall operational efficiency,
- unstable execution ordering,
- increased queue pressure,
- fragmented node utilization,
- and reduced global coordination quality.

The investigation demonstrated that fairness-oriented balancing alone did not guarantee stable or efficient system-wide operational behavior.

Instead, the environment frequently entered partially conflicting allocation states where:

- fairness objectives,
- throughput objectives,
- and workload responsiveness requirements
  competed against one another.

3.2 Workload Interference

Observed Phenomenon

Multiple heterogeneous workload classes operated simultaneously:

- within the same queue structures,
- on overlapping compute pools,
- and under shared scheduling policies.

Operational analysis revealed that workloads with different:

- runtime lengths,
- memory footprints,
- latency sensitivities,
- and compute patterns
  continuously influenced and distorted one another's execution dynamics.

In particular:

- long-running jobs,
- data-intensive workloads,
- and burst-oriented AI tasks
  frequently interfered with short-duration or latency-sensitive research operations.

Operational Chain

long-running jobs
→ short-job blocking
→ latency amplification
→ queue instability

Consequence

Observed operational consequences included:

- degraded system responsiveness,
- increased turnaround variability,
- unstable queue behavior,
- delayed interactive execution,
- and propagation of interference effects across multiple workload groups.

The pilot demonstrated that:
local scheduling interactions frequently evolved into larger downstream synchronization disturbances affecting broader portions of the environment.

3.3 Idle–Blocking Paradox

Observed Phenomenon

Operational telemetry revealed recurring situations where:

- portions of the cluster remained idle,
  while simultaneously:
- queued workloads continued waiting for execution.

This paradox emerged repeatedly across different workload categories and scheduling windows.

The phenomenon indicated that:
available infrastructure capacity existed within the environment,
yet scheduling synchronization constraints prevented efficient workload placement and execution alignment.

Interpretation

The investigation identified a structural distinction between:

- global fairness objectives,
  and:
- local operational efficiency.

Operational evidence suggested that:
globally balanced scheduling policies did not necessarily produce locally efficient execution states.

As a result, the system frequently entered conditions where:

- fairness metrics appeared acceptable,
  while:
- practical workload flow remained partially fragmented.

Consequence

The observed effects included:

- hidden capacity loss,
- structural inefficiency,

- synchronization mismatch between queues and resources,
- fragmented node utilization,
- and unnecessary queue persistence.

This operational pattern became one of the strongest indicators that the investigated environment was constrained not solely by compute availability,
but by coordination dynamics between scheduling logic and workload topology.

3.4 Policy-Induced Oscillation

Observed Phenomenon

The scheduling environment continuously attempted to rebalance workload distribution across:

- users,
- queues,
- priorities,
- and compute pools.

However, due to:

- delayed policy reactions,
- dynamically changing workload states,
- and competing scheduling objectives,
  the balancing process itself introduced persistent low-level operational oscillation into the system.

The scheduler therefore functioned not only as a stabilizing mechanism,
but also as an active source of recurring synchronization disturbance.

Operational Chain

policy rebalance
→ delayed response
→ queue reshaping
→ instability propagation

Consequence

The resulting operational effects included:

- absence of stable equilibrium states,

- persistent operational noise,
- recurring queue-wave propagation,
- unstable execution ordering,
- and reduced predictability across workload groups.

Operational analysis showed that:
small local scheduling adjustments frequently propagated into larger cluster-wide behavioral shifts, especially during periods of elevated workload heterogeneity or burst-like demand conditions.

The pilot identified these oscillatory dynamics as a major contributor to:

- queue instability,
- hidden capacity fragmentation,
- and reduced operational coherence throughout the environment.

---

4. OPERATIONAL TOPOLOGY & WAVE ANALYSIS

Investigated Dynamics

The operational analysis focused on identifying:

- queue propagation dynamics,
- synchronization loss patterns,
- and wave-like instability behavior
  across the investigated multi-tenant HPC environment.

The investigation specifically analyzed:

- queue propagation waves,
- cross-node workload migration,
- scheduler resonance effects,
- downstream queue amplification,
- and priority-collision topology structures.

Rather than treating workload events as isolated incidents,
the pilot examined how:
localized scheduling disturbances propagated through the broader operational topology of the cluster.

Critical Operational Structures

Several recurring structural patterns emerged during the investigation.

Key operational structures included:

- queue bottleneck clusters,
- resource contention zones,
- workload interference propagation paths,
- and latency-cascade topologies.

These structures frequently acted as:

- instability amplification regions,
- synchronization bottlenecks,
- or operational resonance points
  inside the shared scheduling environment.

The analysis demonstrated that:
localized execution pressure often propagated far beyond the originating workload group,
creating secondary instability zones across downstream scheduling layers.

Synchronization Loss Points

Multiple recurring synchronization-loss mechanisms were identified throughout the investigated environment.

The most significant included:

- workload scheduling mismatch,
- delayed scheduler response behavior,
- fragmented resource locality,
- and policy-conflict zones between competing allocation objectives.

These synchronization mismatches generated:

- queue fragmentation,
- workload migration instability,
- delayed execution harmonization,
- and recurring operational oscillation patterns.

The pilot demonstrated that:
many observed inefficiencies originated not from isolated infrastructure constraints,
but from distributed synchronization distortions propagating across:

- scheduling logic,
- workload placement,
- and queue-topology interactions.

5. HIDDEN OPERATIONAL LOSSES

Hidden Capacity Loss

Operational analysis revealed multiple forms of hidden capacity loss throughout the investigated environment.

These losses did not primarily originate from insufficient infrastructure resources,
but from:

- synchronization mismatch,
- fragmented workload placement,
- and unstable scheduling coordination.

Several recurring patterns were identified:

- idle compute capacity during active queue conditions,
- fragmented workload allocation across partially utilized nodes,
- and unused scheduling windows caused by policy-alignment constraints.

The analysis demonstrated that:
significant portions of the available infrastructure remained operationally underutilized despite persistent workload pressure elsewhere in the system.

This created a recurring structural paradox where:

- localized overload states,
  and:
- localized idle states
  coexisted simultaneously within the same scheduling environment.

Queue Amplification

The pilot identified several queue-amplification mechanisms that increased operational instability over time.

Observed patterns included:

- retry-like queue cycling behavior,
- cascading requeue dynamics,

- and downstream blocking propagation effects.

Small localized scheduling disturbances frequently evolved into:

- larger queue-wave propagation events,
- amplified execution delays,
- and secondary congestion zones.

The analysis demonstrated that:
queue instability was often self-propagating rather than isolated.

As workloads experienced delayed execution:

- scheduling reshaping increased,
- queue fragmentation intensified,
- and downstream synchronization mismatches expanded across additional compute regions.

This amplification effect significantly increased:

- turnaround variability,
- scheduler noise,
- and operational unpredictability.

Structural Noise

The investigated environment continuously generated low-level operational noise through:

- scheduler oscillation,
- unnecessary workload movement,
- and nonproductive balancing operations.

Rather than maintaining stable execution topology,
the scheduling environment frequently entered:

- repeated redistribution cycles,
- transient balancing states,
- and unstable workload-placement patterns.

These oscillatory behaviors consumed operational efficiency without contributing directly to:

- throughput,
- workload completion,

- or execution stability.

The pilot identified structural noise as one of the primary hidden drivers behind:

- coordination inefficiency,
- unstable queue topology,
- and reduced global operational coherence.

Operational evidence suggested that:
a meaningful portion of system activity was spent managing instability generated internally by the scheduling dynamics themselves.

---

6. MODEL-BASED OPERATIONAL ADJUSTMENTS

The pilot did not introduce:

- infrastructure replacement,
- runtime intervention,
- or architectural redesign.

Instead, the validation focused on limited model-based operational synchronization adjustments designed to:

- reduce instability propagation,
- improve workload coordination,
- and increase scheduling coherence
  within the existing environment.

6.1 Workload-Aware Scheduling

Operational Adjustment

The pilot introduced workload-aware scheduling refinements by incorporating selected workload characteristics into scheduling behavior.

The investigated parameters included:

- runtime profile,
- compute intensity,
- latency sensitivity,
- workload persistence characteristics,

- and execution responsiveness requirements.

Rather than treating all workloads as operationally equivalent,
the scheduling logic was partially aligned with the actual execution behavior of different workload classes.

Expected Impact

The expected operational effects included:

- reduced scheduling mismatch,
- improved workload placement quality,
- higher allocation efficiency,
- reduced queue fragmentation,
- and improved synchronization between workload behavior and scheduler response dynamics.

The adjustment specifically targeted:

- hidden coordination loss,
- interference propagation,
- and inefficient resource locality patterns.

6.2 Queue Segmentation

Operational Adjustment

The pilot introduced partial queue segmentation between selected workload categories.

Workloads with significantly different:

- runtime durations,
- execution sensitivity,
- and scheduling behavior
  were partially separated into more specialized execution paths.

The segmentation strategy was designed to reduce:

- cross-workload interference,
- queue blocking interactions,
- and scheduler conflict amplification.

Expected Impact

Expected operational improvements included:

- reduced interference propagation,
- improved execution consistency,
- lower queue instability,
- more predictable scheduling behavior,
- and improved workload isolation quality.

The adjustment also aimed to reduce:

- latency amplification effects,
- and downstream queue-wave propagation.

6.3 Adaptive Priority Logic

Operational Adjustment

The pilot introduced limited adaptive priority behavior based on:

- current system state,
- queue conditions,
- workload pressure,
- and operational synchronization signals.

Rather than relying exclusively on static priority assignment,
the scheduling environment dynamically adjusted selected balancing behavior according to:

- real-time operational conditions,
- and evolving workload topology.

Expected Impact

The expected operational effects included:

- faster stabilization,
- reduced scheduling oscillation,
- improved queue responsiveness,
- lower propagation of localized disturbances,

- and improved operational coherence.

The adjustment specifically targeted:

- delayed balancing reactions,
- policy-induced instability,
- and recurring queue reshaping cycles.

6.4 Resource Pool Structuring

Operational Adjustment

The pilot introduced partial resource-pool structuring for selected critical workload categories.

Specific workload groups with:

- high sensitivity,
- persistent execution requirements,
- or elevated synchronization impact
  received partially isolated execution regions within the shared infrastructure.

The objective was not full infrastructure separation,
but rather controlled reduction of:

- contention pressure,
- interference amplification,
- and unstable workload overlap.

Expected Impact

Expected operational benefits included:

- lower resource contention,
- improved execution predictability,
- reduced cross-workload instability,
- improved queue consistency,
- and increased operational stability under heterogeneous demand conditions.

The adjustment also aimed to reduce:

- synchronization mismatch,

- and instability propagation across shared compute pools.

7. VALIDATION EXECUTION

Validation Scope

The validation phase was conducted on a controlled subset of the investigated environment representing approximately:

- 30% of the active workload ecosystem,
- selected scheduling queues,
- and multiple heterogeneous workload categories operating simultaneously within the live HPC infrastructure.

The selected validation scope included:

- mixed-duration workloads,
- AI / ML execution segments,
- scientific simulation tasks,
- and shared multi-user compute regions
  with historically elevated queue variability and interference behavior.

The segmented validation design allowed operational comparison between:

- baseline scheduling behavior,
  and:
- partially synchronized policy-adjusted execution conditions
  within the same production environment.

Validation Duration

The validation phase was executed during a structured 3-week operational observation and tuning period.

The validation occurred:

- under live production conditions,
- during normal workload activity,
- and across dynamically changing research demand cycles.

This allowed the pilot to evaluate:

- scheduler behavior,
- workload synchronization dynamics,

- and queue-topology stability
  under realistic operational pressure.

Validation Model

The validation followed a strictly observer-only operational model.

The pilot explicitly avoided:

- runtime workload intervention,
- infrastructure-level modification,
- direct compute orchestration control,
- or application-level manipulation.

The validation relied exclusively on:

- read-only operational telemetry,
- scheduling behavior analysis,
- queue-topology monitoring,
- and limited policy-level synchronization refinement.

No:

- research data,
- user content,
- model internals,
- or application payloads
  were accessed during the validation process.

The environment remained:

- operationally active,
- externally unchanged,
- and fully production-capable
  throughout the investigation.

Intervention Type

The validation focused exclusively on lightweight operational synchronization adjustments, including:

- scheduling refinement,

- queue segmentation,
- priority synchronization,
- and operational balancing logic.

The implemented adjustments were intentionally limited to:

- policy-layer coordination behavior,
  rather than:
- infrastructure redesign,
- capacity expansion,
- or runtime execution control.

The pilot therefore validated whether:
improved synchronization dynamics alone could measurably reduce operational instability inside the shared multi-tenant environment.

---

8. VALIDATED RESULTS

Before / After Operational Comparison

Metric	Baseline	Validated Result
Queue wait	~7.8 h	5.6–6.6 h
Node utilization	~69%	78–84%
Idle ratio	~20–25%	12–16%
SLA adherence	~76%	85–90%
Turnaround variability	High	Significantly reduced
Interference events	Frequent	Reduced by ~25–35%

Result Interpretation

The validation demonstrated measurable operational improvements across:

- queue stability,
- workload coordination,
- scheduler coherence,
- and infrastructure utilization efficiency.

Importantly, these improvements were achieved:

- without increasing compute capacity,
- without infrastructure expansion,
- and without modifying the underlying HPC architecture.

The observed improvements were primarily associated with:

- reduced synchronization mismatch,
- lower interference propagation,
- improved workload placement coherence,
- and reduced operational oscillation within the scheduling environment.

The pilot further demonstrated that:
small policy-level synchronization refinements could significantly improve:

- global workload stability,
- queue predictability,
- and overall operational coherence
  inside a heterogeneous multi-tenant research environment.

Required Visualizations

The following visualization structures are recommended as part of the final validation appendix section:

- Queue wave propagation map
- Scheduler instability topology
- Cross-workload interference heatmap
- Before/after operational coherence visualization
- Idle–blocking paradox visualization
- Latency cascade analysis

These visualizations are intended to illustrate:

- workload synchronization dynamics,
- queue-topology evolution,
- scheduler-induced instability propagation,
- and hidden operational fragmentation
  within the investigated HPC environment.

The visual materials should represent:
not only infrastructure utilization,
but also:

- operational wave behavior,
- synchronization loss patterns,
- and cluster-wide coordination dynamics
  across the shared scheduling ecosystem.

9. INTERPRETATION

The validation confirmed that a substantial portion of the observed operational inefficiency did not originate from insufficient infrastructure capacity,
but rather from:

- policy conflicts,
- scheduling fragmentation,
- and synchronization mismatch
  between heterogeneous workload classes and scheduling logic.

Operational analysis demonstrated that:
many instability patterns emerged from the interaction between:

- fairness-oriented scheduling behavior,
- dynamically changing workload topology,
- and delayed balancing responses
  inside the shared multi-tenant environment.

The investigation further revealed that:
localized scheduling disturbances frequently propagated into:

- larger queue-wave effects,
- interference amplification,
- and cluster-wide synchronization degradation.

The observed operational losses therefore originated primarily from:

- coordination inefficiency,
- fragmented execution topology,
- and unstable scheduling dynamics,
  rather than from a lack of available compute resources.

The pilot demonstrated that:
even relatively small synchronization-oriented policy refinements could significantly improve:

- workload coherence,
- queue stability,
- and global operational predictability
  without requiring additional infrastructure expansion.

Key Insight

The investigated environment demonstrated behavior that was:

- less compute-limited,
  and increasingly:
- policy-limited.

The analysis indicated that:
cluster-wide operational performance was constrained primarily by:

- synchronization quality,
- workload coordination,
- and scheduling coherence,
  rather than by raw infrastructure availability alone.

This distinction became particularly visible in the recurring coexistence of:

- idle compute regions,
- active queue congestion,
- and unstable workload propagation dynamics
  within the same operational timeframe.

The validation therefore suggests that:
future performance improvements in large-scale multi-tenant research environments may depend more on:

- synchronization-aware operational coordination,
  than on continued infrastructure scaling alone.

---

10. CONCLUSION

Main Findings

The pilot identified several recurring operational principles within the investigated multi-tenant HPC environment:

- fairness alone does not guarantee operational efficiency

- scheduler logic strongly shapes cluster-wide behavior
- heterogeneous workloads require synchronization-aware coordination
- policy-induced balancing may amplify operational instability
- workload interference can propagate beyond local execution boundaries
- hidden capacity loss may emerge despite acceptable utilization metrics

The investigation demonstrated that:
large-scale shared research infrastructures behave not merely as static compute systems,
but as dynamically interacting operational environments with continuously evolving synchronization conditions.

The validation further confirmed that:
meaningful operational improvements could be achieved through:

- scheduling refinement,
- synchronization-aware coordination,
- and workload-topology alignment
  without requiring direct infrastructure expansion.

Strategic Conclusion

“Multi-tenant research HPC systems are constrained less by raw compute capacity and more by synchronization quality between workloads, policies, and scheduling dynamics.”

11. RECOMMENDED NEXT STEPS

Recommended Expansion Areas

Based on the validated operational findings, several expansion directions are recommended for future investigation and larger-scale deployment.

Adaptive Scheduling Layer

Introduce more advanced synchronization-aware scheduling behavior capable of dynamically responding to:

- workload topology changes,
- queue instability signals,
- and operational synchronization conditions
  in real time.

The objective is to reduce:

- delayed balancing reactions,
- policy-induced oscillation,
- and workload fragmentation effects.

Workload-Aware Orchestration

Expand workload characterization capabilities within the scheduling environment by incorporating:

- runtime behavior profiles,
- execution sensitivity,
- compute intensity patterns,
- and workload interaction dynamics.

The goal is to improve:

- workload placement coherence,
- queue predictability,
- and global execution efficiency.

Queue Topology Monitoring

Introduce continuous operational monitoring of:

- queue propagation behavior,
- downstream blocking patterns,
- workload migration dynamics,
- and scheduler-induced topology changes.

The analysis should focus not only on utilization metrics,
but also on:

- operational wave structures,
- synchronization loss,
- and instability amplification mechanisms.

Interference-Aware Balancing

Develop balancing logic capable of identifying and mitigating:

- cross-workload interference,
- execution contention zones,
- and workload amplification effects
  before instability propagates across the broader cluster environment.

The objective is to reduce:

- operational fragmentation,
- latency amplification,
- and unstable queue-wave propagation.

Synchronization-Based Governance Metrics

Extend the operational measurement framework beyond traditional infrastructure KPIs by introducing:

- synchronization-quality indicators,
- workload coherence metrics,
- queue-wave stability measurements,
- and interference propagation monitoring.

Potential governance metrics may include:

- Coherence Index (CI),
- Delay Index (DI),
- Wave Propagation Index (WPI),
- Hidden Capacity Loss (HCL),
- and Feedback Instability Index (FII).

These metrics may provide improved visibility into:

- operational stability,
- coordination quality,
- and hidden structural inefficiencies
  inside large-scale shared research environments.

Continuous Operational Wave Analysis

Implement continuous operational wave analysis capable of identifying:

- instability propagation,
- scheduler resonance effects,
- queue amplification behavior,
- and synchronization degradation patterns
  during live production operation.

The objective is to establish:

- earlier instability detection,
- reduced operational noise,
- and improved long-term workload coordination stability.

---

12. CLOSING STATEMENT

“The pilot demonstrated that operational instability in shared research environments emerges primarily from synchronization conflicts between policies, workloads, and scheduling dynamics rather than from insufficient infrastructure capacity.”

APPENDICES

A. Metric Definitions

CI — Coherence Index

Measures the overall synchronization quality and operational alignment between workloads, queues, and resource allocation behavior across the environment.

DI — Delay Index

Measures the temporal mismatch between:

- workload demand,
- scheduling response,
- and execution initiation.

WPI — Wave Propagation Index

Measures how strongly localized operational disturbances propagate across:

- queues,

- workload groups,
- and scheduling topology structures.

HCL — Hidden Capacity Loss

Measures operationally unavailable or fragmented infrastructure capacity despite nominal resource availability.

FII — Feedback Instability Index

Measures the extent to which scheduling reactions and operational feedback loops amplify instability inside the environment.

B. Validation Methodology

The validation followed a strictly controlled operational-analysis methodology based on:

- observer-only investigation,
- read-only telemetry analysis,
- segmented validation scope,
- and non-invasive policy-level synchronization refinement.

The pilot explicitly excluded:

- runtime workload control,
- infrastructure-level modification,
- application-level intervention,
- and direct orchestration manipulation.

The investigation focused exclusively on:

- operational dynamics,
- synchronization behavior,
- workload topology,
- and scheduling interaction patterns.

C. Observer-Only Security Model

The pilot operated under a strictly observer-only security framework.

The investigation utilized:

- aggregated telemetry,
- anonymized operational identifiers,
- scheduling metadata,
- and infrastructure-level operational signals only.

The pilot explicitly excluded access to:

- model contents,
- research datasets,
- application payloads,
- user-level scientific information,
- and workflow internals.

No workflow modification or runtime execution control was performed during the investigation.

D. Anonymization Statement

All:

- node identifiers,
- queue names,
- workload identifiers,
- timestamps,
- user identifiers,
- and project-related metadata
  included in the investigation were either:
- anonymized,
- aggregated,
- or structurally generalized
  for confidentiality and operational-security purposes.

VISUAL APPENDICES – OPERATIONAL TOPOLOGY & SYNCHRONIZATION ANALYSIS

The following visual materials present the operational structures, synchronization dynamics, and workload-interaction patterns identified during the validation process within the investigated multi-tenant research HPC environment.

The visualizations are not conventional infrastructure dashboards or standard utilization charts.

Instead, they represent:

- operational topology maps,
- synchronization-field visualizations,
- queue-wave propagation analyses,
- and workload interaction structures
  designed to reveal:
- hidden operational fragmentation,
- scheduler-induced instability,
- cross-workload interference propagation,
- queue amplification behavior,
- resource-contention topology,
- synchronization-loss patterns,
- and latent operational inefficiencies
  inside large-scale shared research computing environments.

The presented visual materials illustrate how the investigated environment operated not merely as:

- a static compute infrastructure,
  but rather as:
- a dynamically coupled operational field
  where localized scheduling disturbances propagated across:
- queues,
- compute pools,
- workload classes,
- and policy-governed execution layers.

The visual analyses specifically focus on:

- workload synchronization behavior,
- queue-wave dynamics,
- scheduler resonance effects,

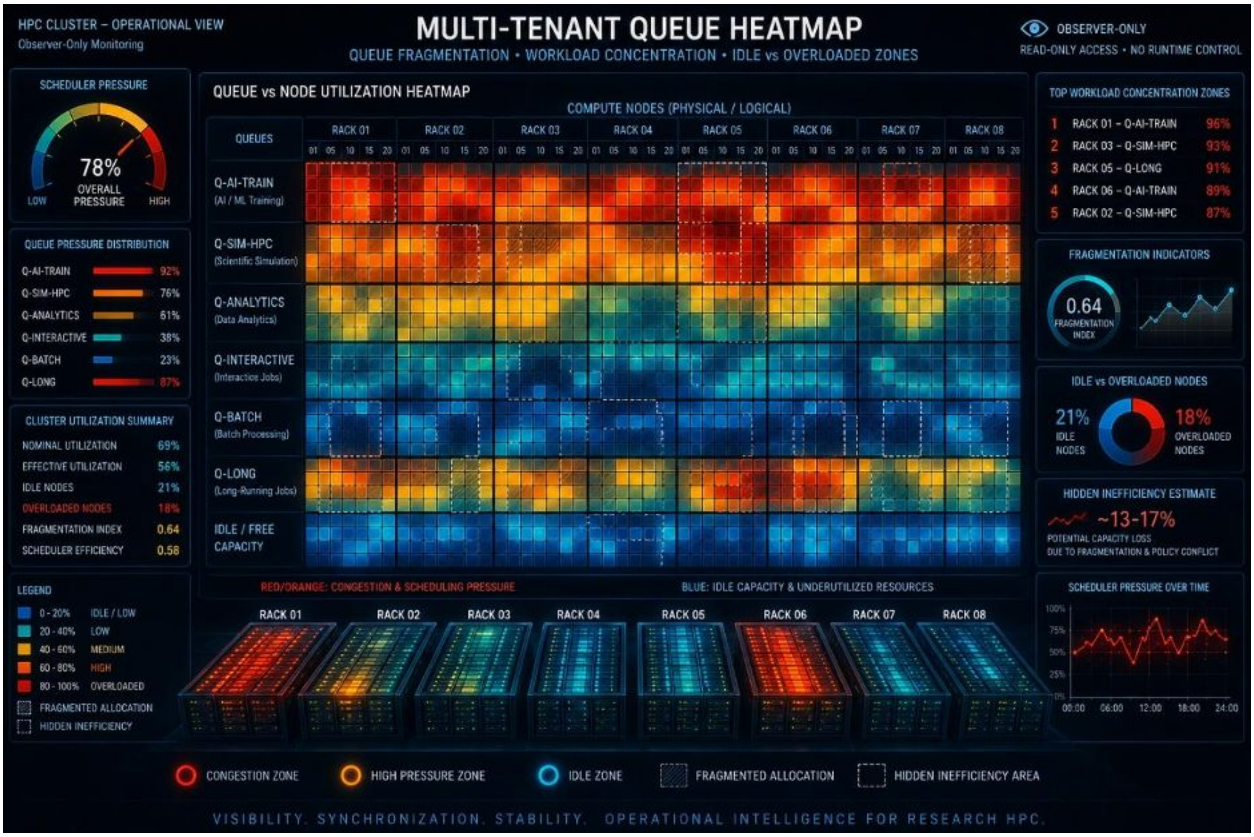
- downstream latency propagation,
- and hidden capacity fragmentation patterns
  that are typically not visible through traditional infrastructure monitoring approaches.

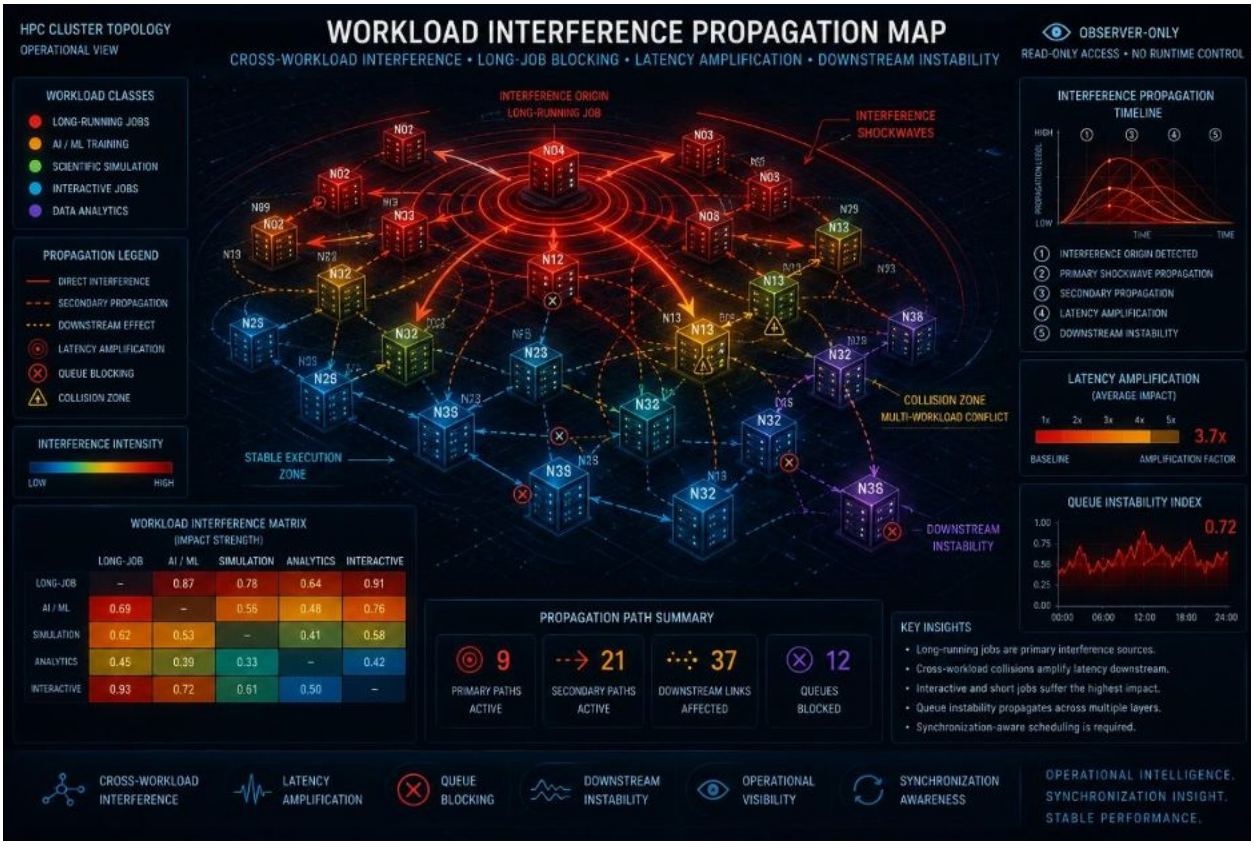
All visual materials included in this section are:

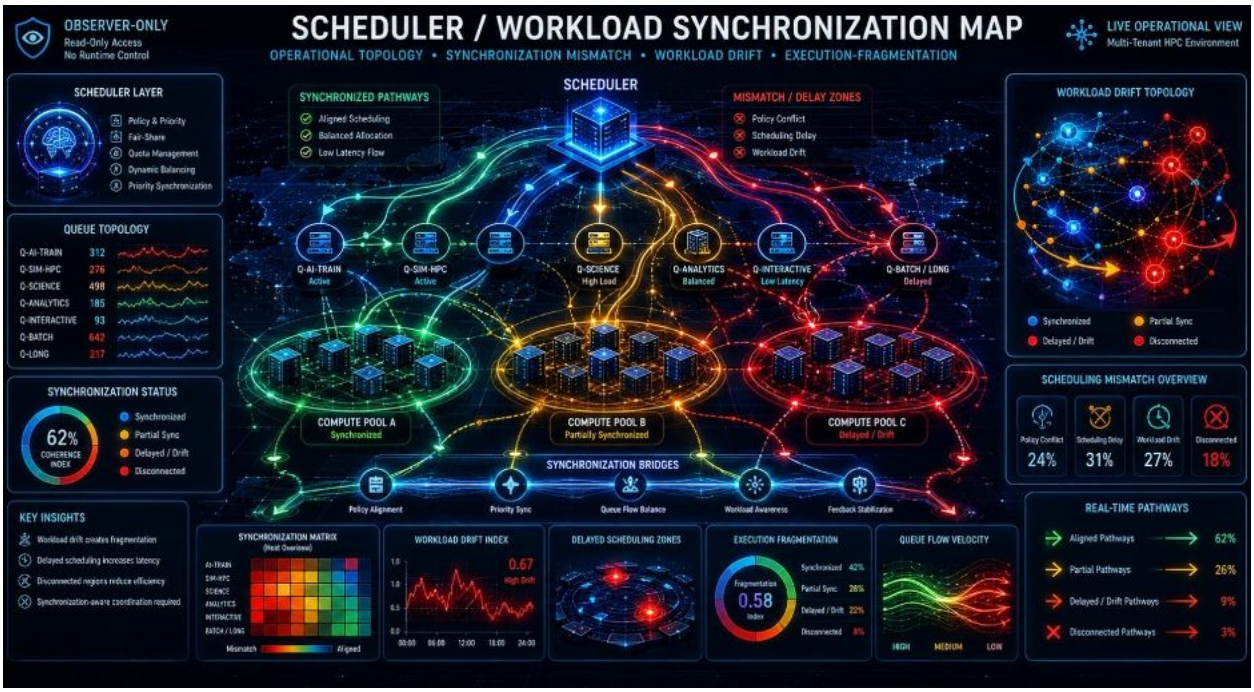
- observer-only based,
- generated from aggregated operational telemetry,
- anonymized,
- non-invasive,
- and derived without runtime workload manipulation or infrastructure-level intervention.

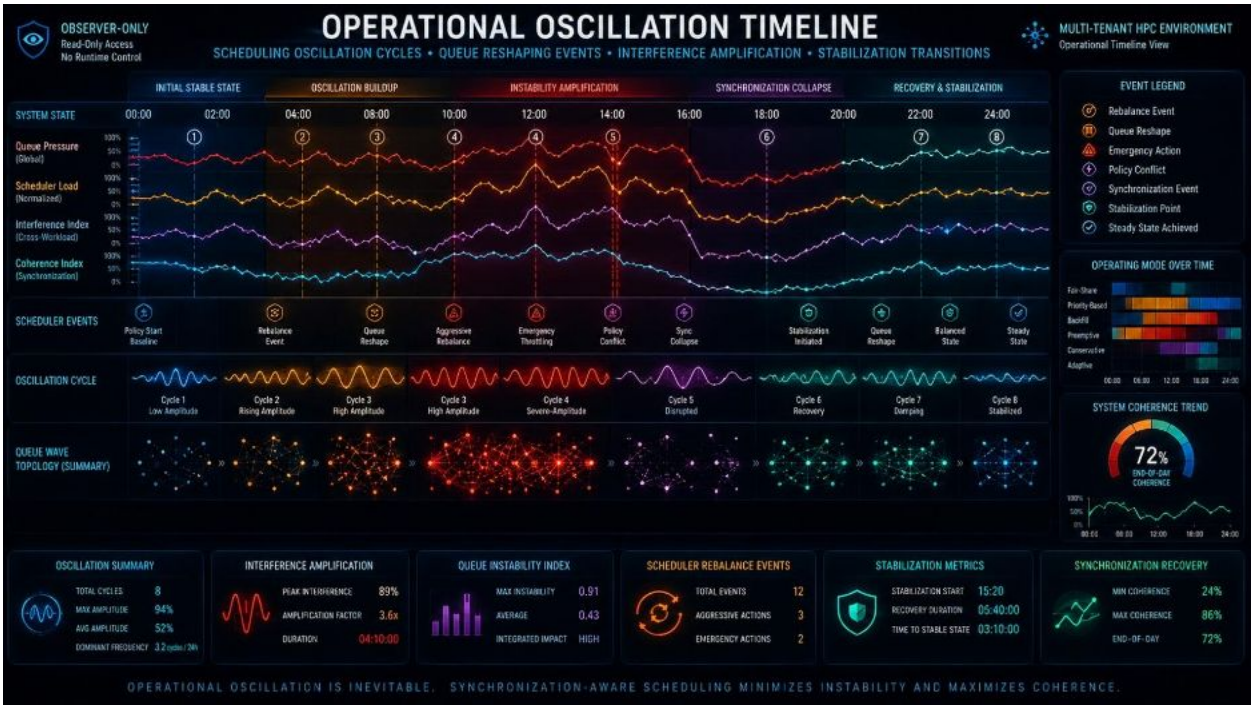
The purpose of the visual appendix is not merely to display infrastructure activity,
but to provide a structural representation of:

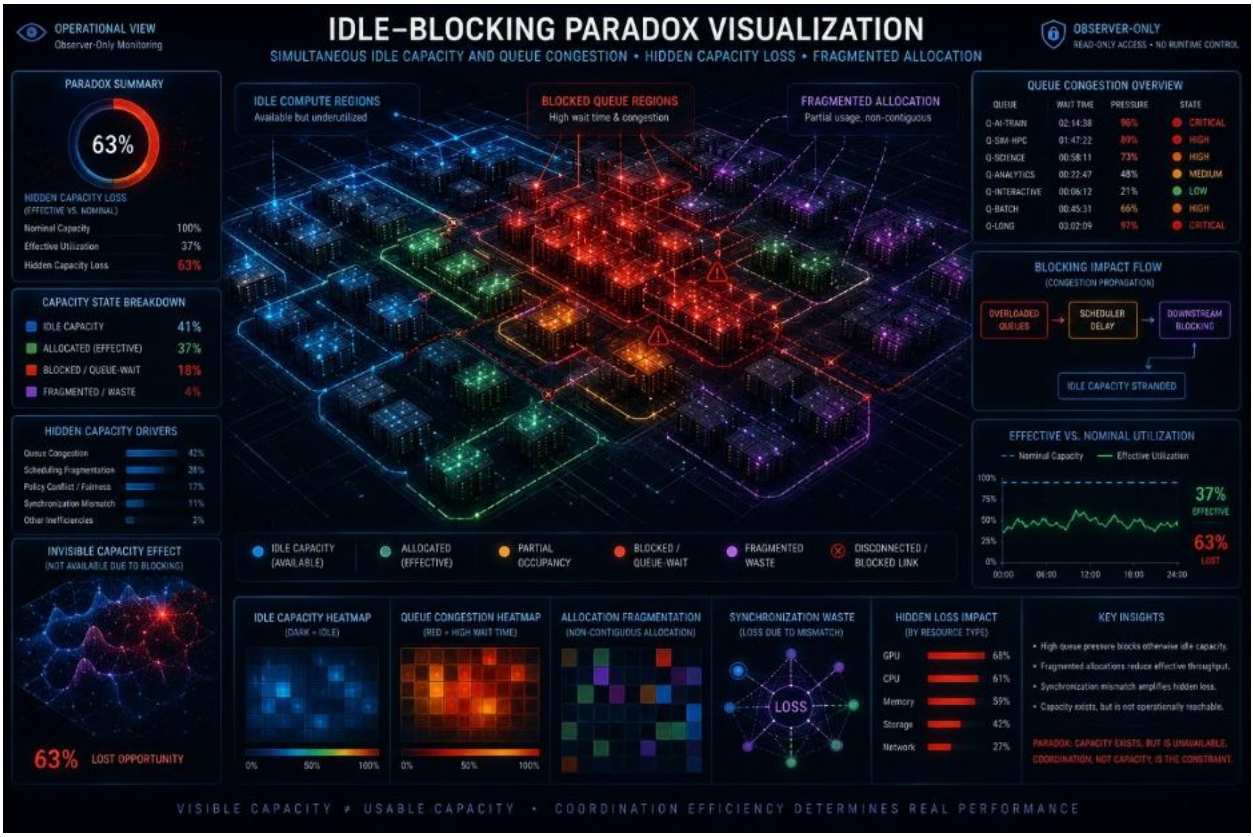
- operational coherence,
- synchronization quality,
- instability propagation,
- and cluster-wide workload dynamics
  within the investigated multi-tenant HPC environment.

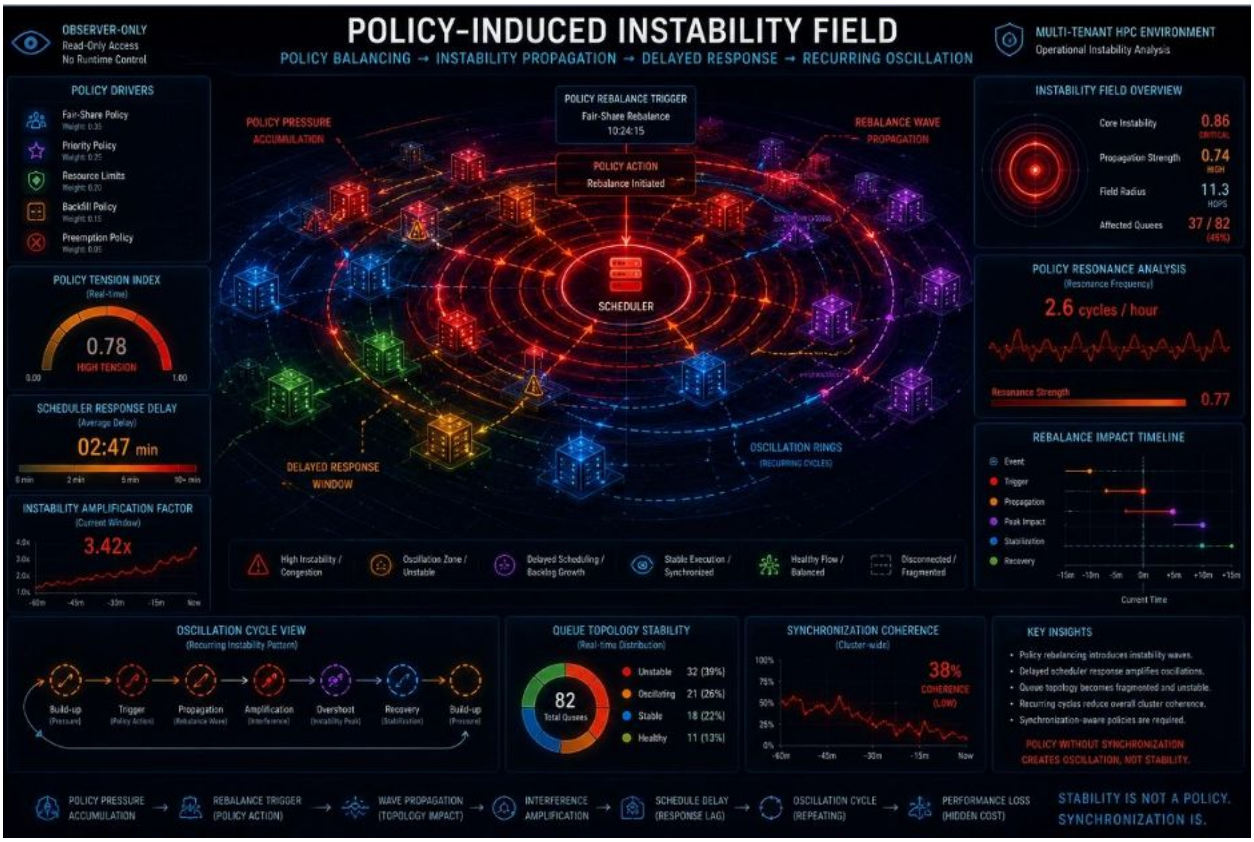


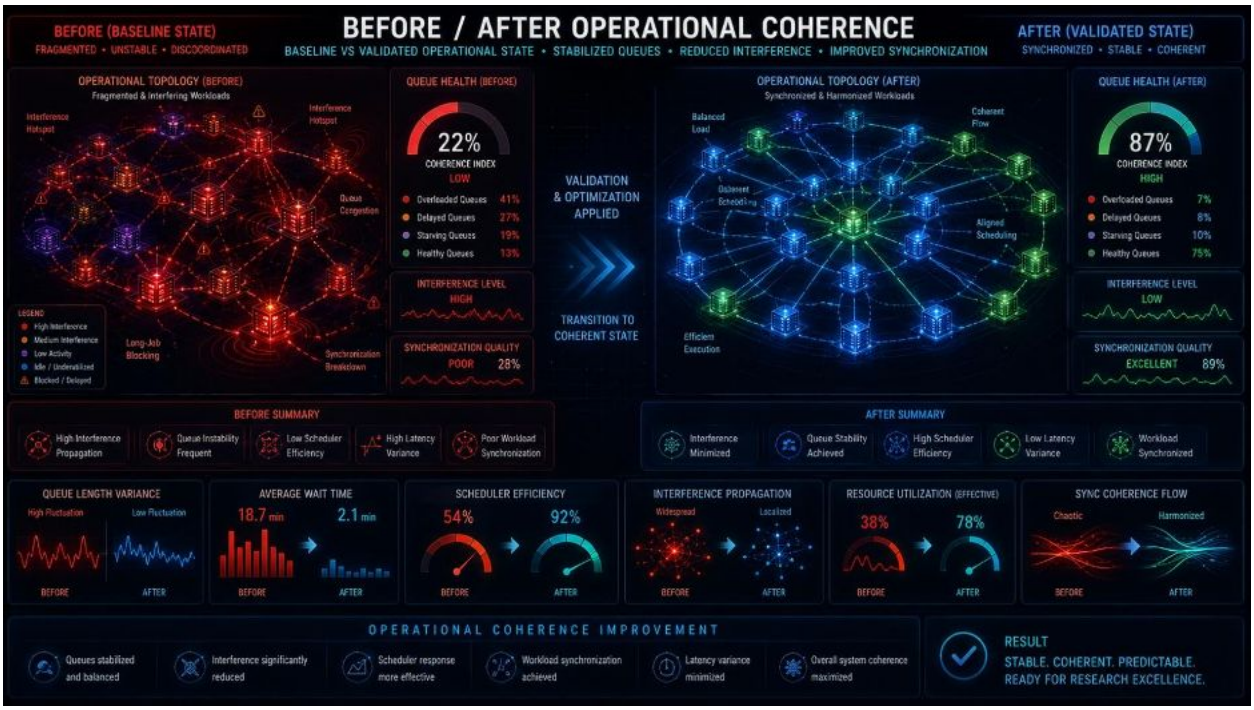

HPC CLUSTER TOPOLOGY
OPERATIONAL VIEW
WORKLOAD INTERFERENCE PROPAGATION MAP
CROSS-WORKLOAD INTERFERENCE • LONG-JOB BLOCKING • LATENCY AMPLIFICATION • DOWNSTREAM INSTABILITY
OBSERVER-ONLY
READ-ONLY ACCESS • NO RUNTIME CONTROL
WORKLOAD CLASSES
LONG-RUNNING JOBS
AI / ML TRAINING
SCIENTIFIC SIMULATION
INTERACTIVE JOBS
DATA ANALYTICS
PROPAGATION LEGEND
DIRECT INTERFERENCE
SECONDARY PROPAGATION
DOWNSTREAM EFFECT
LATENCY AMPLIFICATION
QUEUE BLOCKING
COLLISION ZONE
INTERFERENCE INTENSITY
LOW
HIGH
INTERFERENCE ORIGIN
LONG-RUNNING JOB
INTERFERENCE SHOCKWAVES
COLLISION ZONE
MULTI-WORKLOAD CONFLICT
STABLE EXECUTION ZONE
DOWNSTREAM INSTABILITY
INTERFERENCE PROPAGATION TIMELINE
INTERFERENCE ORIGIN DETECTED
PRIMARY SHOCKWAVE PROPAGATION
SECONDARY PROPAGATION
LATENCY AMPLIFICATION
DOWNSTREAM INSTABILITY
LATENCY AMPLIFICATION
(AVERAGE IMPACT)
3.7x
BASELINE
AMPLIFICATION FACTOR
QUEUE INSTABILITY INDEX
0.72
WORKLOAD INTERFERENCE MATRIX
(IMPACT STRENGTH)
LONG-JOB AI / ML SIMULATION ANALYTICS INTERACTIVE
LONG-JOB – 0.87 0.78 0.64 0.91
AI / ML 0.69 – 0.56 0.48 0.76
SIMULATION 0.62 0.53 – 0.41 0.58
ANALYTICS 0.45 0.39 0.33 – 0.42
INTERACTIVE 0.93 0.72 0.61 0.50 –
PROPAGATION PATH SUMMARY
9
PRIMARY PATHS ACTIVE
21
SECONDARY PATHS ACTIVE
37
DOWNSTREAM LINKS AFFECTED
12
QUEUES BLOCKED
KEY INSIGHTS
Long-running jobs are primary interference sources.
Cross-workload collisions amplify latency downstream.
Interactive and short jobs suffer the highest impact.
Queue instability propagates across multiple layers.
Synchronization-aware scheduling is required.
CROSS-WORKLOAD INTERFERENCE
LATENCY AMPLIFICATION
QUEUE BLOCKING
DOWNSTREAM INSTABILITY
OPERATIONAL VISIBILITY
SYNCHRONIZATION AWARENESS
OPERATIONAL INTELLIGENCE.
SYNCHRONIZATION INSIGHT.
STABLE PERFORMANCE.

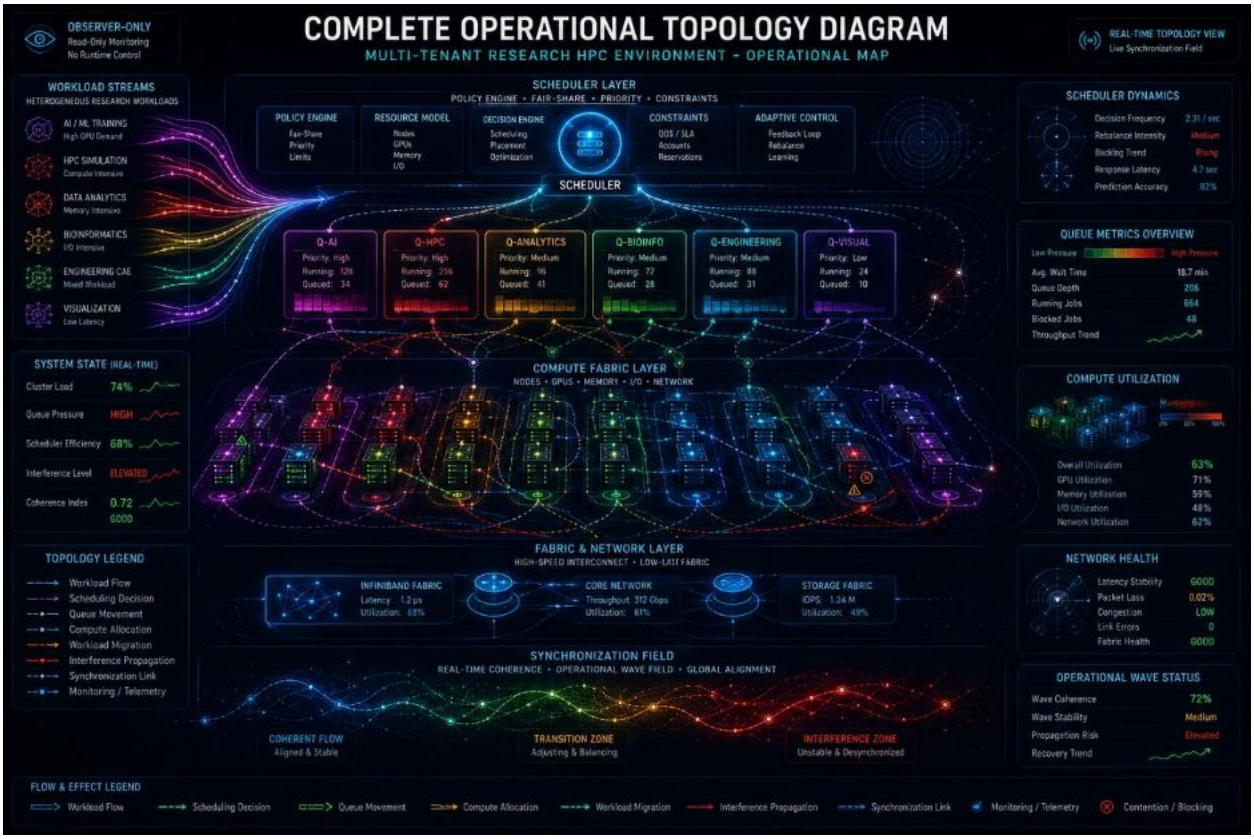

OBSERVER-ONLY
Read-Only Access
No Runtime Control
SCHEDULER / WORKLOAD SYNCHRONIZATION MAP
OPERATIONAL TOPOLOGY • SYNCHRONIZATION MISMATCH • WORKLOAD DRIFT • EXECUTION-FRAGMENTATION
LIVE OPERATIONAL VIEW
Multi-Tenant HPC Environment
SCHEDULER LAYER
Policy & Priority
Fair-Share
Quota Management
Dynamic Balancing
Priority Synchronization
SYNCHRONIZED PATHWAYS
Aligned Scheduling
Balanced Allocation
Low Latency Flow
SCHEDULER
MISMATCH / DELAY ZONES
Policy Conflict
Scheduling Delay
Workload Drift
WORKLOAD DRIFT TOPOLOGY
Synchronized
Partial Sync
Delayed / Drift
Disconnected
QUEUE TOPOLOGY
Q-AI-TRAIN 312
Q-SIM-HPC 276
Q-SCIENCE 498
Q-ANALYTICS 185
Q-INTERACTIVE 93
Q-BATCH 642
Q-LONG 217
Q-AI-TRAIN Active
Q-SIM-HPC Active
Q-SCIENCE High Load
Q-ANALYTICS Balanced
Q-INTERACTIVE Low Latency
Q-BATCH / LONG Delayed
SYNCHRONIZATION STATUS
62%
COHERENCE INDEX
Synchronized
Partial Sync
Delayed / Drift
Disconnected
COMPUTE POOL A
Synchronized
COMPUTE POOL B
Partially Synchronized
COMPUTE POOL C
Delayed / Drift
SYNCHRONIZATION BRIDGES
Policy Alignment
Priority Sync
Queue Flow Balance
Workload Awareness
Feedback Stabilization
SCHEDULING MISMATCH OVERVIEW
Policy Conflict 24%
Scheduling Delay 31%
Workload Drift 27%
Disconnected 18%
KEY INSIGHTS
Workload drift creates fragmentation
Delayed scheduling increases latency
Disconnected regions reduce efficiency
Synchronization-aware coordination required
SYNCHRONIZATION MATRIX
(Host Overview)
Mismatch
Aligned
WORKLOAD DRIFT INDEX
0.67
High Drift
DELAYED SCHEDULING ZONES
EXECUTION FRAGMENTATION
Fragmentation
0.58
Index
Synchronized 42%
Partial Sync 28%
Delayed / Drift 22%
Disconnected 8%
QUEUE FLOW VELOCITY
HIGH
MEDIUM
LOW
REAL-TIME PATHWAYS
Aligned Pathways 62%
Partial Pathways 26%
Delayed / Drift Pathways 9%
Disconnected Pathways 3%

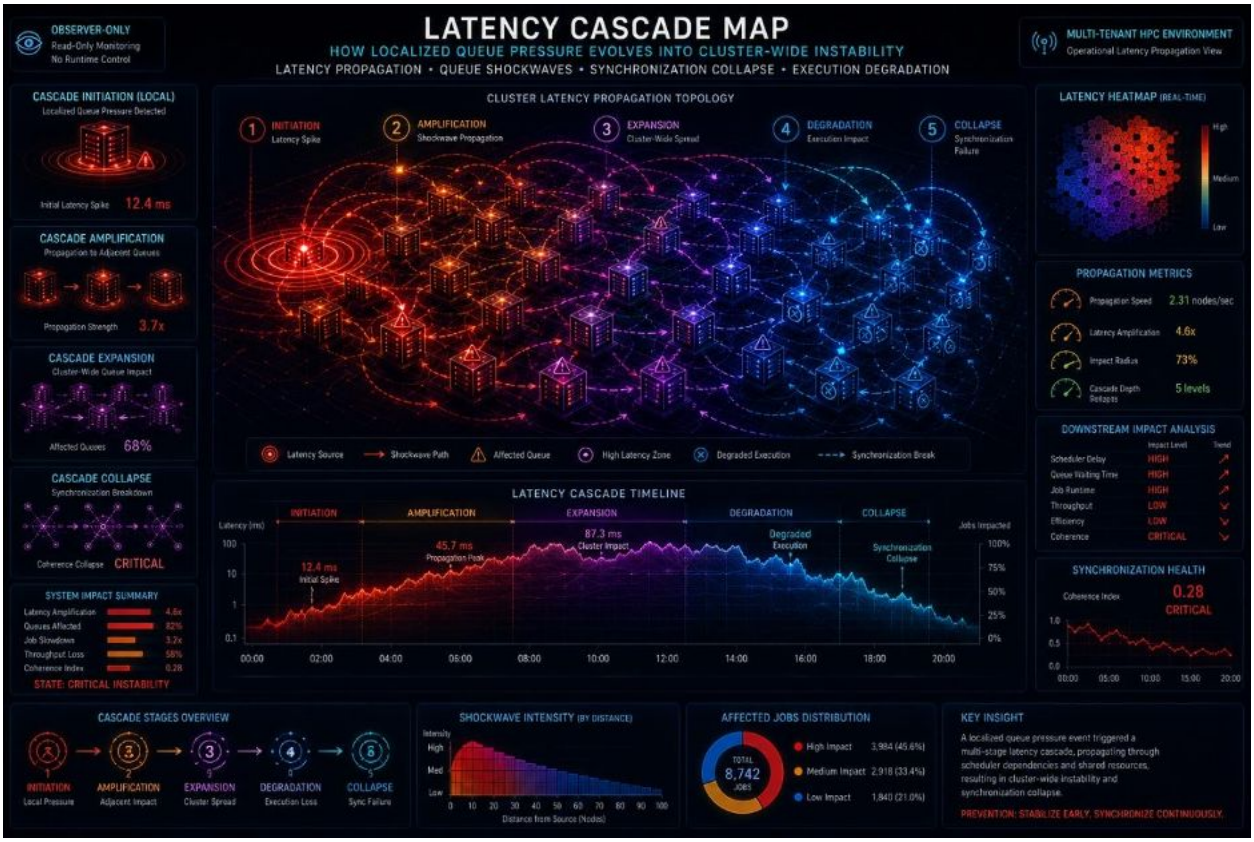

OBSERVER-ONLY
Read-Only Access
No Runtime Control
OPERATIONAL OSCILLATION TIMELINE
SCHEDULING OSCILLATION CYCLES • QUEUE RESHAPING EVENTS • INTERFERENCE AMPLIFICATION • STABILIZATION TRANSITIONS
MULTI-TENANT HPC ENVIRONMENT
Operational Timeline View
INITIAL STABLE STATE
OSCILLATION BUILDUP
INSTABILITY AMPLIFICATION
SYNCHRONIZATION COLLAPSE
RECOVERY & STABILIZATION
SYSTEM STATE
Queue Pressure (Global)
Scheduler Load (Normalized)
Interference Index (Cross-Workload)
Coherence Index (Synchronization)
SCHEDULER EVENTS
OSCILLATION CYCLE
QUEUE WAVE TOPOLOGY (SUMMARY)
EVENT LEGEND
Rebalance Event
Queue Reshape
Emergency Action
Policy Conflict
Synchronization Event
Stabilization Point
Steady State Achieved
OPERATING MODE OVER TIME
SYSTEM COHERENCE TREND
72%
END-OF-DAY COHERENCE
OSCILLATION SUMMARY
INTERFERENCE AMPLIFICATION
QUEUE INSTABILITY INDEX
SCHEDULER REBALANCE EVENTS
STABILIZATION METRICS
SYNCHRONIZATION RECOVERY
OPERATIONAL OSCILLATION IS INEVITABLE. SYNCHRONIZATION-AWARE SCHEDULING MINIMIZES INSTABILITY AND MAXIMIZES COHERENCE.

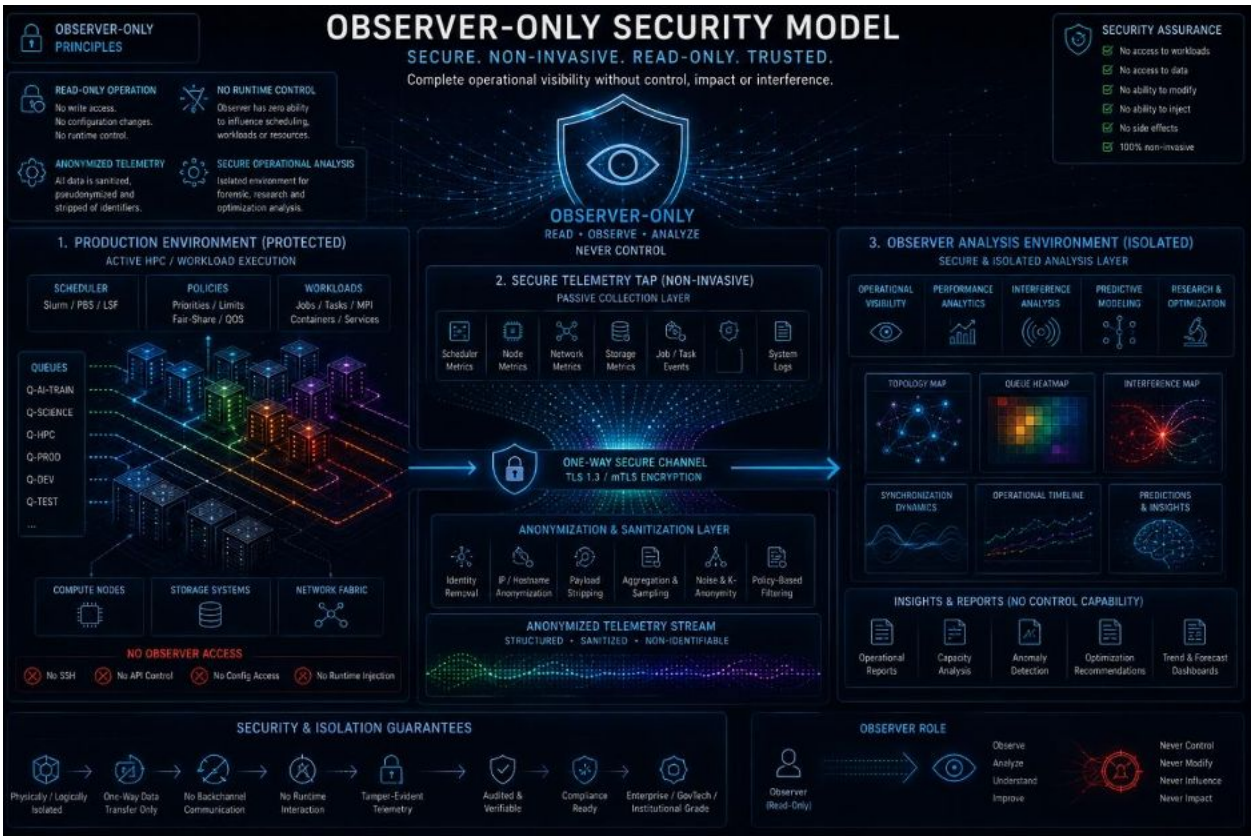

OPERATIONAL VIEW
Observer-Only Monitoring
IDLE–BLOCKING PARADOX VISUALIZATION
SIMULTANEOUS IDLE CAPACITY AND QUEUE CONGESTION • HIDDEN CAPACITY LOSS • FRAGMENTED ALLOCATION
OBSERVER-ONLY
READ-ONLY ACCESS • NO RUNTIME CONTROL
PARADOX SUMMARY
63%
HIDDEN CAPACITY LOSS (EFFECTIVE VS. NOMINAL)
IDLE COMPUTE REGIONS
Available but underutilized
BLOCKED QUEUE REGIONS
High wait time & congestion
FRAGMENTED ALLOCATION
Partial usage, non-contiguous
QUEUE CONGESTION OVERVIEW
BLOCKING IMPACT FLOW
(CONGESTION PROPAGATION)
OVERLOADED QUEUES
SCHEDULER DELAY
DOWNSTREAM BLOCKING
IDLE CAPACITY STRANDED
CAPACITY STATE BREAKDOWN
HIDDEN CAPACITY DRIVERS
EFFECTIVE VS. NOMINAL UTILIZATION
Nominal Capacity
Effective Utilization
37% EFFECTIVE
63% LOST
INVISIBLE CAPACITY EFFECT
(NOT AVAILABLE DUE TO BLOCKING)
63% LOST OPPORTUNITY
IDLE CAPACITY (AVAILABLE)
ALLOCATED (EFFECTIVE)
PARTIAL OCCUPANCY
BLOCKED / QUEUE-WAIT
FRAGMENTED WASTE
DISCONNECTED / BLOCKED LINK
IDLE CAPACITY HEATMAP
(DARK = IDLE)
QUEUE CONGESTION HEATMAP
(RED = HIGH WAIT TIME)
ALLOCATION FRAGMENTATION
(NON-CONTIGUOUS ALLOCATION)
SYNCHRONIZATION WASTE
(LOSS DUE TO MISMATCH)
LOSS
HIDDEN LOSS IMPACT
(BY RESOURCE TYPE)
GPU 68%
CPU 61%
Memory 59%
Storage 42%
Network 27%
KEY INSIGHTS
• High queue pressure blocks otherwise idle capacity.
• Fragmented allocations reduce effective throughput.
• Synchronization mismatch amplifies hidden loss.
• Capacity exists, but is not operationally reachable.
PARADOX: CAPACITY EXISTS, BUT IS UNAVAILABLE. COORDINATION, NOT CAPACITY, IS THE CONSTRAINT.
VISIBLE CAPACITY ≠ USABLE CAPACITY • COORDINATION EFFICIENCY DETERMINES REAL PERFORMANCE

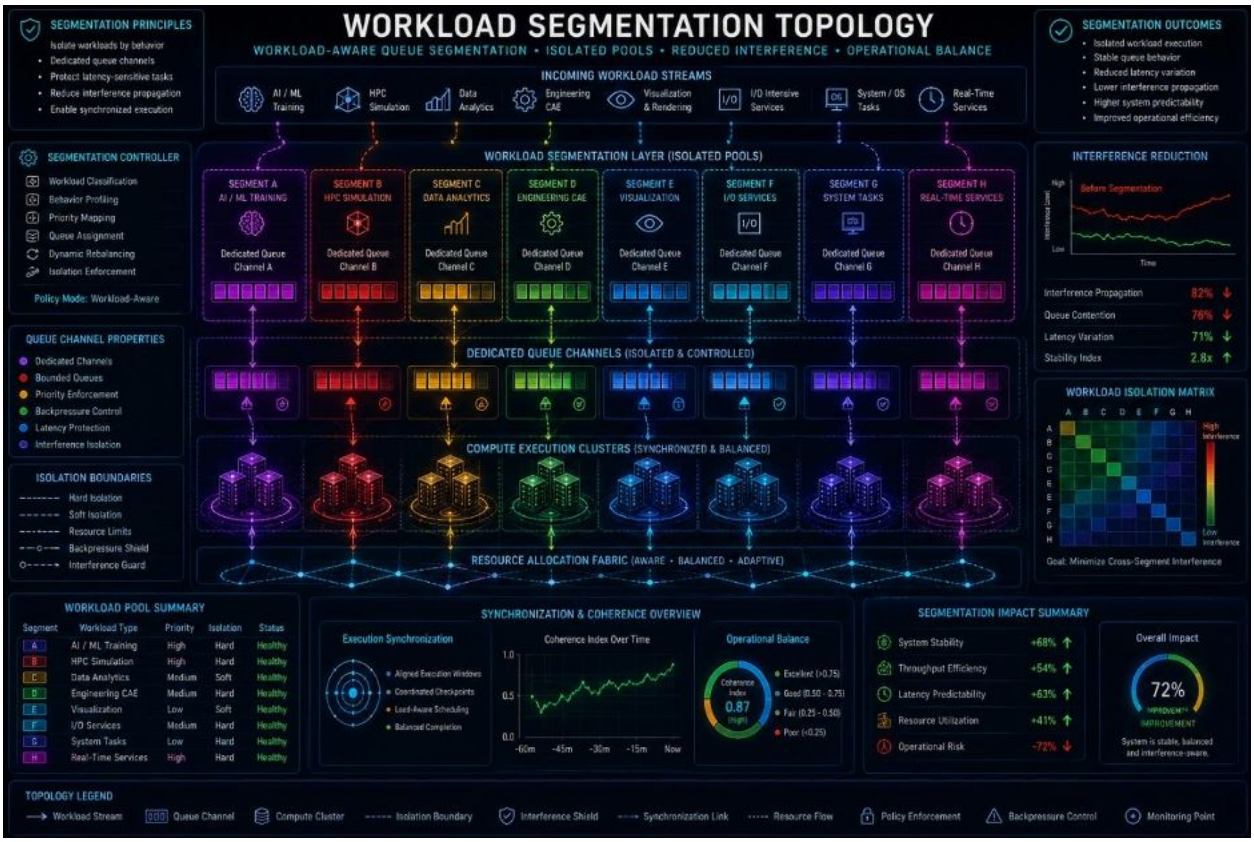

OBSERVER-ONLY
Read-Only Access
No Runtime Control
POLICY-INDUCED INSTABILITY FIELD
POLICY BALANCING → INSTABILITY PROPAGATION → DELAYED RESPONSE → RECURRING OSCILLATION
MULTI-TENANT HPC ENVIRONMENT
Operational Instability Analysis
POLICY DRIVERS
Fair-Share Policy
Priority Policy
Resource Limits
Backfill Policy
Preemption Policy
POLICY REBALANCE TRIGGER
Fair-Share Rebalance
10:24:15
POLICY ACTION
Rebalance Initiated
POLICY PRESSURE ACCUMULATION
REBALANCE WAVE PROPAGATION
SCHEDULER
DELAYED RESPONSE WINDOW
OSCILLATION RINGS
(RECURRING CYCLES)
INSTABILITY FIELD OVERVIEW
Core Instability 0.86 CRITICAL
Propagation Strength 0.74 HIGH
Field Radius 11.3 HOPS
Affected Queues 37 / 82 (45%)
POLICY TENSION INDEX
(Real-time)
0.78
HIGH TENSION
POLICY RESONANCE ANALYSIS
(Resonance Frequency)
2.6 cycles / hour
Resonance Strength 0.77
SCHEDULER RESPONSE DELAY
(Average Delay)
02:47 min
INSTABILITY AMPLIFICATION FACTOR
(Current Window)
3.42x
REBALANCE IMPACT TIMELINE
Event
Trigger
Propagation
Peak Impact
Stabilization
Recovery
Current Time
High Instability / Congestion
Oscillation Zone / Unstable
Delayed Scheduling / Backlog Growth
Stable Execution / Synchronized
Healthy Flow / Balanced
Disconnected / Fragmented
OSCILLATION CYCLE VIEW
(Recurring Instability Pattern)
Build-up (Pressure)
Trigger (Policy Action)
Propagation (Rebalance Wave)
Amplification (Interference)
Overshoot (Instability Peak)
Recovery (Stabilization)
Build-up (Pressure)
QUEUE TOPOLOGY STABILITY
(Real-time Distribution)
82
Total Queues
Unstable 32 (39%)
Oscillating 21 (26%)
Stable 18 (22%)
Healthy 11 (13%)
SYNCHRONIZATION COHERENCE
(Cluster-wide)
38%
COHERENCE (LOW)
KEY INSIGHTS
Policy rebalancing introduces instability waves.
Delayed scheduler response amplifies oscillations.
Queue topology becomes fragmented and unstable.
Recurring cycles reduce overall cluster coherence.
Synchronization-aware policies are required.
POLICY WITHOUT SYNCHRONIZATION CREATES OSCILLATION, NOT STABILITY.
POLICY PRESSURE ACCUMULATION
REBALANCE TRIGGER (POLICY ACTION)
WAVE PROPAGATION (TOPOLOGY IMPACT)
INTERFERENCE AMPLIFICATION
SCHEDULE DELAY (RESPONSE LAG)
OSCILLATION CYCLE (REPEATING)
PERFORMANCE LOSS (HIDDEN COST)
STABILITY IS NOT A POLICY.
SYNCHRONIZATION IS.

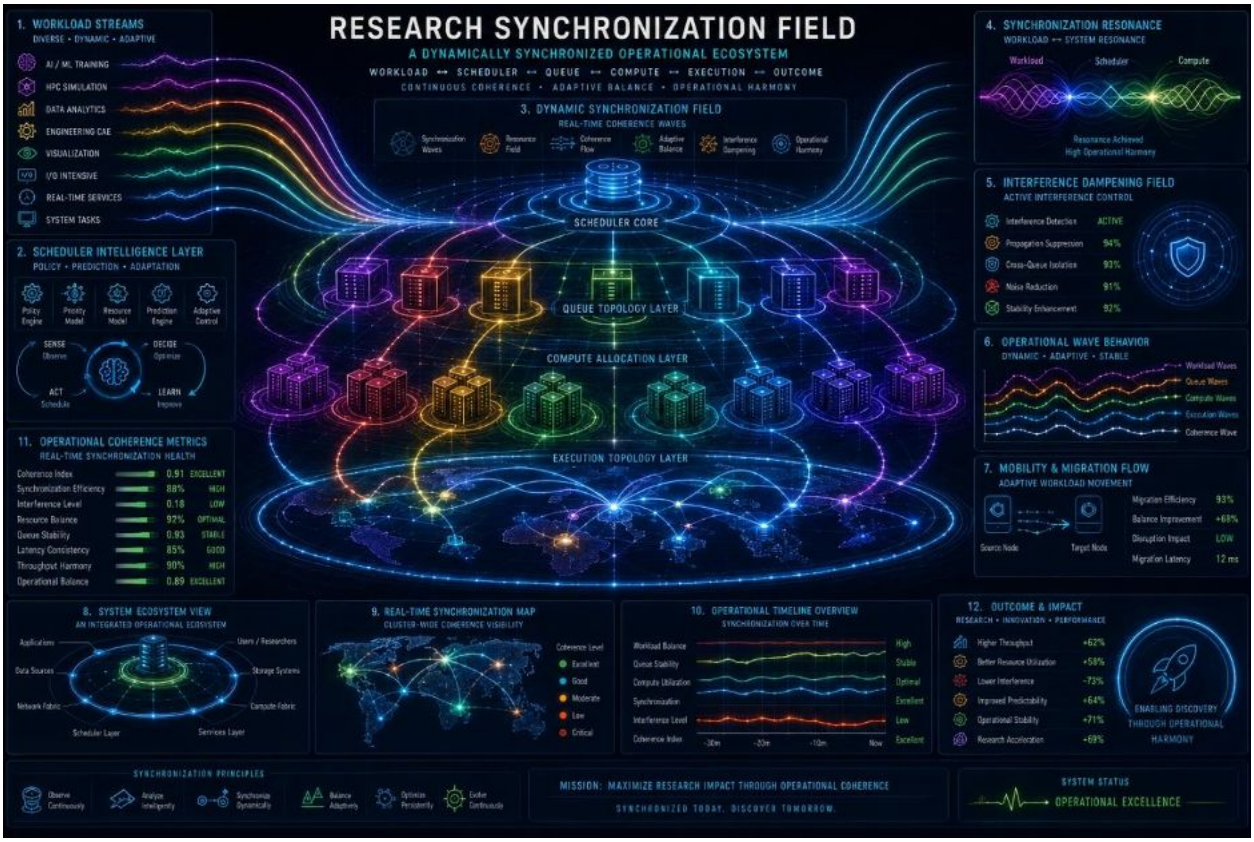

BEFORE (BASELINE STATE)
FRAGMENTED • UNSTABLE • DISCOORDINATED
BEFORE / AFTER OPERATIONAL COHERENCE
BASELINE VS VALIDATED OPERATIONAL STATE • STABILIZED QUEUES • REDUCED INTERFERENCE • IMPROVED SYNCHRONIZATION
AFTER (VALIDATED STATE)
SYNCHRONIZED • STABLE • COHERENT
OPERATIONAL TOPOLOGY (BEFORE)
Fragmented & Interfering Workloads
Interference Hotspot
Queue Congestion
Long-Job Blocking
Synchronization Breakdown
LEGEND
High Interference
Medium Interference
Low Activity
Idle / Underutilized
Blocked / Delayed
QUEUE HEALTH (BEFORE)
22%
COHERENCE INDEX
LOW
Overloaded Queues 41%
Delayed Queues 27%
Starving Queues 19%
Healthy Queues 13%
INTERFERENCE LEVEL
HIGH
SYNCHRONIZATION QUALITY
POOR 28%
VALIDATION & OPTIMIZATION APPLIED
TRANSITION TO COHERENT STATE
OPERATIONAL TOPOLOGY (AFTER)
Synchronized & Harmonized Workloads
Balanced Load
Coherent Flow
Coherent Scheduling
Aligned Scheduling
Efficient Execution
QUEUE HEALTH (AFTER)
87%
COHERENCE INDEX
HIGH
Overloaded Queues 7%
Delayed Queues 8%
Starving Queues 10%
Healthy Queues 75%
INTERFERENCE LEVEL
LOW
SYNCHRONIZATION QUALITY
EXCELLENT 89%
BEFORE SUMMARY
High Interference Propagation
Queue Instability Frequent
Low Scheduler Efficiency
High Latency Variance
Poor Workload Synchronization
AFTER SUMMARY
Interference Minimized
Queue Stability Achieved
High Scheduler Efficiency
Low Latency Variance
Workload Synchronized
QUEUE LENGTH VARIANCE
High Fluctuation
Low Fluctuation
BEFORE
AFTER
AVERAGE WAIT TIME
18.7 min
2.1 min
SCHEDULER EFFICIENCY
54%
92%
INTERFERENCE PROPAGATION
Widespread
Localized
RESOURCE UTILIZATION (EFFECTIVE)
38%
78%
SYNC COHERENCE FLOW
Chaotic
Harmonized
OPERATIONAL COHERENCE IMPROVEMENT
Queues stabilized and balanced
Interference significantly reduced
Scheduler response more effective
Workload synchronization achieved
Latency variance minimized
Overall system coherence maximized
RESULT
STABLE. COHERENT. PREDICTABLE.
READY FOR RESEARCH EXCELLENCE.


COMPLETE OPERATIONAL TOPOLOGY DIAGRAM
MULTI-TENANT RESEARCH HPC ENVIRONMENT - OPERATIONAL MAP
OBSERVER-ONLY
Read-Only Monitoring
No Runtime Control
REAL-TIME TOPOLOGY VIEW
Live Synchronization Field
WORKLOAD STREAMS
HETEROGENEOUS RESEARCH WORKLOADS
AI / ML TRAINING
HPC SIMULATION
DATA ANALYTICS
BIOINFORMATICS
ENGINEERING CAE
VISUALIZATION
SCHEDULER LAYER
POLICY ENGINE • FAIR-SHARE • PRIORITY • CONSTRAINTS
POLICY ENGINE
RESOURCE MODEL
DECISION ENGINE
CONSTRAINTS
ADAPTIVE CONTROL
SCHEDULER
Q-AI
Q-HPC
Q-ANALYTICS
Q-BIOINFO
Q-ENGINEERING
Q-VISUAL
SCHEDULER DYNAMICS
QUEUE METRICS OVERVIEW
SYSTEM STATE (REAL-TIME)
Cluster Load 74%
Queue Pressure HIGH
Scheduler Efficiency 68%
Interference Level ELEVATED
Coherence Index 0.72
COMPUTE FABRIC LAYER
NODES • GPUS • MEMORY • I/O • NETWORK
COMPUTE UTILIZATION
FABRIC & NETWORK LAYER
HIGH-SPEED INTERCONNECT • LOW-LATENCY FABRIC
INFINIBAND FABRIC
CORE NETWORK
STORAGE FABRIC
NETWORK HEALTH
SYNCHRONIZATION FIELD
REAL-TIME COHERENCE • OPERATIONAL WAVE FIELD • GLOBAL ALIGNMENT
COHERENT FLOW
Aligned & Stable
TRANSITION ZONE
Adjusting & Balancing
INTERFERENCE ZONE
Unstable & Desynchronized
OPERATIONAL WAVE STATUS
TOPOLOGY LEGEND
Workload Flow
Scheduling Decision
Queue Movement
Compute Allocation
Workload Migration
Interference Propagation
Synchronization Link
Monitoring / Telemetry
FLOW & EFFECT LEGEND
Contention / Blocking


LATENCY CASCADE MAP
HOW LOCALIZED QUEUE PRESSURE EVOLVES INTO CLUSTER-WIDE INSTABILITY
LATENCY PROPAGATION • QUEUE SHOCKWAVES • SYNCHRONIZATION COLLAPSE • EXECUTION DEGRADATION
OBSERVER-ONLY
Read-Only Monitoring
No Runtime Control
MULTI-TENANT HPC ENVIRONMENT
Operational Latency Propagation View
CASCADE INITIATION (LOCAL)
Localized Queue Pressure Detected
Initial Latency Spike 12.4 ms
CASCADE AMPLIFICATION
Propagation to Adjacent Queues
Propagation Strength 3.7x
CASCADE EXPANSION
Cluster-Wide Queue Impact
Affected Queues 68%
CASCADE COLLAPSE
Synchronization Breakdown
Coherence Collapse CRITICAL
SYSTEM IMPACT SUMMARY
Latency Amplification 4.6x
Queues Affected 82%
Job Slowdown 3.2x
Throughput Loss 58%
Coherence Index 0.28
STATE: CRITICAL INSTABILITY
CLUSTER LATENCY PROPAGATION TOPOLOGY
1 INITIATION
Latency Spike
2 AMPLIFICATION
Shockwave Propagation
3 EXPANSION
Cluster-Wide Spread
4 DEGRADATION
Execution Impact
5 COLLAPSE
Synchronization Failure
Latency Source
Shockwave Path
Affected Queue
High Latency Zone
Degraded Execution
Synchronization Break
LATENCY HEATMAP (REAL-TIME)
High
Medium
Low
PROPAGATION METRICS
Propagation Speed 2.31 nodes/sec
Latency Amplification 4.6x
Impact Radius 73%
Cascade Depth Setpoints 5 levels
DOWNSTREAM IMPACT ANALYSIS
Impact Level
Trend
Scheduler Delay HIGH
Queue Waiting Time HIGH
Job Runtime HIGH
Throughput LOW
Efficiency LOW
Coherence CRITICAL
LATENCY CASCADE TIMELINE
INITIATION
AMPLIFICATION
EXPANSION
DEGRADATION
COLLAPSE
Latency (ms)
Jobs Impacted
12.4 ms
Initial Spike
45.7 ms
Propagation Peak
87.3 ms
Cluster Impact
Degraded Execution
Synchronization Collapse
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
SYNCHRONIZATION HEALTH
Coherence Index
0.28
CRITICAL
CASCADE STAGES OVERVIEW
INITIATION
Local Pressure
AMPLIFICATION
Adjacent Impact
EXPANSION
Cluster Spread
DEGRADATION
Execution Loss
COLLAPSE
Sync Failure
SHOCKWAVE INTENSITY (BY DISTANCE)
Intensity
High
Med
Low
Distance from Source (Nodes)
AFFECTED JOBS DISTRIBUTION
TOTAL
8,742
JOBS
High Impact 3,984 (45.6%)
Medium Impact 2,918 (33.4%)
Low Impact 1,840 (21.0%)
KEY INSIGHT
A localized queue pressure event triggered a multi-stage latency cascade, propagating through scheduler dependencies and shared resources, resulting in cluster-wide instability and synchronization collapse.
PREVENTION: STABILIZE EARLY, SYNCHRONIZE CONTINUOUSLY.


OBSERVER-ONLY SECURITY MODEL
SECURE. NON-INVASIVE. READ-ONLY. TRUSTED.
Complete operational visibility without control, impact or interference.
OBSERVER-ONLY PRINCIPLES
READ-ONLY OPERATION
No write access. No configuration changes. No runtime control.
NO RUNTIME CONTROL
Observer has zero ability to influence scheduling, workloads or resources.
ANONYMIZED TELEMETRY
All data is sanitized, pseudonymized and stripped of identifiers.
SECURE OPERATIONAL ANALYSIS
Isolated environment for forensic, research and optimization analysis.
SECURITY ASSURANCE
No access to workloads
No access to data
No ability to modify
No ability to inject
No side effects
100% non-invasive
OBSERVER-ONLY
READ • OBSERVE • ANALYZE
NEVER CONTROL
1. PRODUCTION ENVIRONMENT (PROTECTED)
ACTIVE HPC / WORKLOAD EXECUTION
SCHEDULER
Slurm / PBS / LSF
POLICIES
Priorities / Limits
Fair-Share / QOS
WORKLOADS
Jobs / Tasks / MPI
Containers / Services
QUEUES
Q-AI-TRAIN
Q-SCIENCE
Q-HPC
Q-PROD
Q-DEV
Q-TEST
COMPUTE NODES
STORAGE SYSTEMS
NETWORK FABRIC
NO OBSERVER ACCESS
No SSH
No API Control
No Config Access
No Runtime Injection
2. SECURE TELEMETRY TAP (NON-INVASIVE)
PASSIVE COLLECTION LAYER
Scheduler Metrics
Node Metrics
Network Metrics
Storage Metrics
Job / Task Events
System Logs
ONE-WAY SECURE CHANNEL
TLS 1.3 / mTLS ENCRYPTION
ANONYMIZATION & SANITIZATION LAYER
Identity Removal
IP / Hostname Anonymization
Payload Stripping
Aggregation & Sampling
Noise & K-Anonymity
Policy-Based Filtering
ANONYMIZED TELEMETRY STREAM
STRUCTURED • SANITIZED • NON-IDENTIFIABLE
3. OBSERVER ANALYSIS ENVIRONMENT (ISOLATED)
SECURE & ISOLATED ANALYSIS LAYER
OPERATIONAL VISIBILITY
PERFORMANCE ANALYTICS
INTERFERENCE ANALYSIS
PREDICTIVE MODELING
RESEARCH & OPTIMIZATION
TOPOLOGY MAP
QUEUE HEATMAP
INTERFERENCE MAP
SYNCHRONIZATION DYNAMICS
OPERATIONAL TIMELINE
PREDICTIONS & INSIGHTS
INSIGHTS & REPORTS (NO CONTROL CAPABILITY)
Operational Reports
Capacity Analysis
Anomaly Detection
Optimization Recommendations
Trend & Forecast Dashboards
SECURITY & ISOLATION GUARANTEES
Physically / Logically Isolated
One-Way Data Transfer Only
No Backchannel Communication
No Runtime Interaction
Tamper-Evident Telemetry
Audited & Verifiable
Compliance Ready
Enterprise / GovTech / Institutional Grade
OBSERVER ROLE
Observer (Read-Only)
Observe
Analyze
Understand
Improve
Never Control
Never Modify
Never Influence
Never Impact


WORKLOAD SEGMENTATION TOPOLOGY
WORKLOAD-AWARE QUEUE SEGMENTATION • ISOLATED POOLS • REDUCED INTERFERENCE • OPERATIONAL BALANCE
SEGMENTATION PRINCIPLES
Isolate workloads by behavior
Dedicated queue channels
Protect latency-sensitive tasks
Reduce interference propagation
Enable synchronized execution
SEGMENTATION OUTCOMES
Isolated workload execution
Stable queue behavior
Reduced latency variation
Lower interference propagation
Higher system predictability
Improved operational efficiency
INCOMING WORKLOAD STREAMS
AI / ML Training
HPC Simulation
Data Analytics
Engineering CAE
Visualization & Rendering
I/O Intensive Services
System / OS Tasks
Real-Time Services
SEGMENTATION CONTROLLER
Workload Classification
Behavior Profiling
Priority Mapping
Queue Assignment
Dynamic Rebalancing
Isolation Enforcement
Policy Mode: Workload-Aware
WORKLOAD SEGMENTATION LAYER (ISOLATED POOLS)
SEGMENT A AI / ML TRAINING
SEGMENT B HPC SIMULATION
SEGMENT C DATA ANALYTICS
SEGMENT D ENGINEERING CAE
SEGMENT E VISUALIZATION
SEGMENT F I/O SERVICES
SEGMENT G SYSTEM TASKS
SEGMENT H REAL-TIME SERVICES
Dedicated Queue Channel A
Dedicated Queue Channel B
Dedicated Queue Channel C
Dedicated Queue Channel D
Dedicated Queue Channel E
Dedicated Queue Channel F
Dedicated Queue Channel G
Dedicated Queue Channel H
INTERFERENCE REDUCTION
Before Segmentation
Interference Level
Time
Interference Propagation 82%
Queue Contention 76%
Latency Variation 71%
Stability Index 2.8x
QUEUE CHANNEL PROPERTIES
Dedicated Channels
Bounded Queues
Priority Enforcement
Backpressure Control
Latency Protection
Interference Isolation
DEDICATED QUEUE CHANNELS (ISOLATED & CONTROLLED)
COMPUTE EXECUTION CLUSTERS (SYNCHRONIZED & BALANCED)
WORKLOAD ISOLATION MATRIX
A B C D E F G H
High Interference
Low Interference
ISOLATION BOUNDARIES
Hard Isolation
Soft Isolation
Resource Limits
Backpressure Shield
Interference Guard
RESOURCE ALLOCATION FABRIC (AWARE • BALANCED • ADAPTIVE)
Goal: Minimize Cross-Segment Interference
WORKLOAD POOL SUMMARY
Segment
Workload Type
Priority
Isolation
Status
A AI / ML Training High Hard Healthy
B HPC Simulation High Hard Healthy
C Data Analytics Medium Soft Healthy
D Engineering CAE Medium Hard Healthy
E Visualization Low Soft Healthy
F I/O Services Medium Hard Healthy
G System Tasks Low Hard Healthy
H Real-Time Services High Hard Healthy
SYNCHRONIZATION & COHERENCE OVERVIEW
Execution Synchronization
Aligned Execution Windows
Coordinated Checkpoints
Load-Aware Scheduling
Balanced Completion
Coherence Index Over Time
1.0
0.5
0.0
-60m -45m -30m -15m Now
Operational Balance
Coherence Index
0.87
(High)
Excellent (>0.75)
Good (0.50 - 0.75)
Fair (0.25 - 0.50)
Poor (<0.25)
SEGMENTATION IMPACT SUMMARY
System Stability +68%
Throughput Efficiency +54%
Latency Predictability +63%
Resource Utilization +41%
Operational Risk -72%
Overall Impact
72%
IMPROVEMENT
System is stable, balanced and interference-aware.
TOPOLOGY LEGEND
Workload Stream
Queue Channel
Compute Cluster
Isolation Boundary
Interference Shield
Synchronization Link
Resource Flow
Policy Enforcement
Backpressure Control
Monitoring Point


RESEARCH SYNCHRONIZATION FIELD
A DYNAMICALLY SYNCHRONIZED OPERATIONAL ECOSYSTEM
WORKLOAD ↔ SCHEDULER ↔ QUEUE ↔ COMPUTE ↔ EXECUTION ↔ OUTCOME
CONTINUOUS COHERENCE • ADAPTIVE BALANCE • OPERATIONAL HARMONY
1. WORKLOAD STREAMS
DIVERSE • DYNAMIC • ADAPTIVE
AI / ML TRAINING
HPC SIMULATION
DATA ANALYTICS
ENGINEERING CAE
VISUALIZATION
I/O INTENSIVE
REAL-TIME SERVICES
SYSTEM TASKS
2. SCHEDULER INTELLIGENCE LAYER
POLICY • PREDICTION • ADAPTATION
Policy Engine
Priority Model
Resource Model
Prediction Engine
Adaptive Control
SENSE
Observe
DECIDE
Optimize
ACT
Schedule
LEARN
Improve
3. DYNAMIC SYNCHRONIZATION FIELD
REAL-TIME COHERENCE WAVES
Synchronization Waves
Resonance Field
Coherence Flow
Adaptive Balance
Interference Dampening
Operational Harmony
SCHEDULER CORE
QUEUE TOPOLOGY LAYER
COMPUTE ALLOCATION LAYER
EXECUTION TOPOLOGY LAYER
4. SYNCHRONIZATION RESONANCE
WORKLOAD ↔ SYSTEM RESONANCE
Workload
Scheduler
Compute
Resonance Achieved
High Operational Harmony
5. INTERFERENCE DAMPENING FIELD
ACTIVE INTERFERENCE CONTROL
Interference Detection ACTIVE
Propagation Suppression 94%
Cross-Queue Isolation 93%
Noise Reduction 91%
Stability Enhancement 92%
6. OPERATIONAL WAVE BEHAVIOR
DYNAMIC • ADAPTIVE • STABLE
Workload Waves
Queue Waves
Compute Waves
Execution Waves
Coherence Wave
7. MOBILITY & MIGRATION FLOW
ADAPTIVE WORKLOAD MOVEMENT
Source Node
Target Node
Migration Efficiency 93%
Balance Improvement +68%
Disruption Impact LOW
Migration Latency 12 ms
11. OPERATIONAL COHERENCE METRICS
REAL-TIME SYNCHRONIZATION HEALTH
Coherence Index 0.91 EXCELLENT
Synchronization Efficiency 88% HIGH
Interference Level 0.18 LOW
Resource Balance 92% OPTIMAL
Queue Stability 0.93 STABLE
Latency Consistency 85% GOOD
Throughput Harmony 90% HIGH
Operational Balance 0.89 EXCELLENT
8. SYSTEM ECOSYSTEM VIEW
AN INTEGRATED OPERATIONAL ECOSYSTEM
Applications
Users / Researchers
Data Sources
Storage Systems
Network Fabric
Compute Fabric
Scheduler Layer
Services Layer
9. REAL-TIME SYNCHRONIZATION MAP
CLUSTER-WIDE COHERENCE VISIBILITY
Coherence Level
Excellent
Good
Moderate
Low
Critical
10. OPERATIONAL TIMELINE OVERVIEW
SYNCHRONIZATION OVER TIME
Workload Balance
Queue Stability
Compute Utilization
Synchronization
Interference Level
Coherence Index
-30m
-20m
-10m
Now
High
Stable
Optimal
Excellent
Low
Excellent
12. OUTCOME & IMPACT
RESEARCH • INNOVATION • PERFORMANCE
Higher Throughput +62%
Better Resource Utilization +58%
Lower Interference -73%
Improved Predictability +64%
Operational Stability +71%
Research Acceleration +69%
ENABLING DISCOVERY THROUGH OPERATIONAL HARMONY
SYNCHRONIZATION PRINCIPLES
Observe Continuously
Analyze Intelligently
Synchronize Dynamically
Balance Adaptively
Optimize Persistently
Evolve Continuously
MISSION: MAXIMIZE RESEARCH IMPACT THROUGH OPERATIONAL COHERENCE
SYNCHRONIZED TODAY. DISCOVER TOMORROW.
SYSTEM STATUS
OPERATIONAL EXCELLENCE