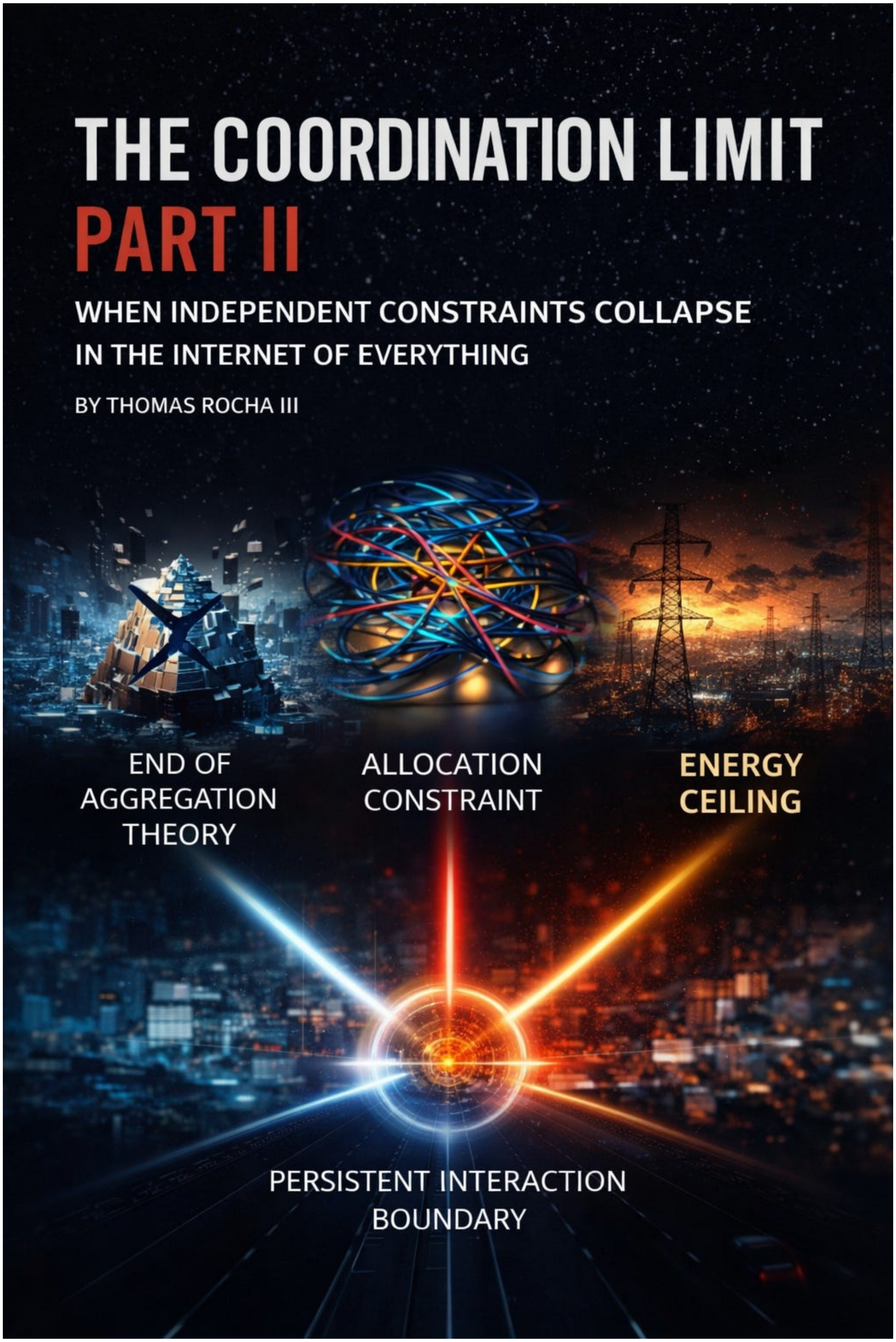
The Coordination Limit, Part II

When Independent Constraints Collapse in the Internet of Everything

THOMAS ROCHA III

APR 18, 2026


THE COORDINATION LIMIT
PART II
WHEN INDEPENDENT CONSTRAINTS COLLAPSE
IN THE INTERNET OF EVERYTHING
BY THOMAS ROCHA III
END OF AGGREGATION THEORY
ALLOCATION CONSTRAINT
ENERGY CEILING
PERSISTENT INTERACTION BOUNDARY

Coordination cost in distributed systems does not scale linearly. It multiplies.

That is the condition I laid out in the last piece.

This is the part people try to engineer their way around.

Three observers have already described the surface of it.

In January 2025, Doug O'Laughlin at Fabricated Knowledge declared that o1 and reasoning models marked the end of Aggregation Theory. Marginal costs had returned to technology. The zero-marginal-cost foundation of the internet era no longer held for AI.

In April 2026, Ben Thompson sharpened the frame. The constraint facing hyperscalers is not marginal cost. It is opportunity cost of compute. Microsoft had the capacity. It had the demand. It chose which workload to serve.

At GTC in March 2026, Jensen Huang described the ceiling in industrial terms. Output is constrained by energy and efficiency, tokens per watt against available gigawatts.

All three are correct. All three describe the system from the outside.

None of them explain what the system consumes while it runs.

The Instinct

The response to this condition has been consistent.

Optimize the system.
Scale the compute.

Abstract the complexity.
Add orchestration.

These are rational responses. They have worked for decades.

They do not work here.

Optimization Does Not Change Class

If coordination cost were linear, optimization would solve the problem.

Reduce overhead. Improve efficiency. Eliminate waste.

But the system is not linear.

It is governed by a multiplicative function:

$$C = k \cdot P \cdot M \cdot F \cdot A \cdot T$$

Participants, modalities, features, authority domains, and transport boundaries do not add cost.

They expand the number of states that must be reconciled during execution.

Optimization reduces the constant.
It does not change the function.

You can make the product smaller.
You cannot make it a sum.

Scaling Compute Makes It Worse

Huang’s framing is correct. It also exposes the trap.

Coordination consumes the same energy pool as computation.

Every additional unit of compute:

- enables more features
- increases interaction complexity
- expands the number of coordination surfaces

The system does not become more efficient.

It becomes more dimensionally complex.

The product grows faster than the capacity.

Scaling compute does not outrun coordination.
It feeds it.

This is what Thompson was seeing from the allocation side. When a hyperscaler chooses which workload to serve, the choice is not between computation and idleness. It is between computation and the coordination overhead of the workloads it did not pick.

Opportunity cost at that scale is not a pricing artifact.

It is the shape of a system where coordination has begun to compete with the thing it was built to support.

The Modularity Reversal

Fragmentation was not a mistake.

It was an advantage.

For two decades, distributed systems were built by separating concerns:

- identity
- transport
- policy
- computation
- orchestration

This allowed:

- independent scaling
- fault isolation
- modular development

The trade-off was coordination overhead.

At low dimensionality, the trade was worth making. The overhead was real but bounded. The benefits of modular development outweighed the cost of reconciling across boundaries.

That trade-off has reversed.

As independent constraints converge, coordination cost exceeds the benefits of modularity.

The same separation that once enabled scale now multiplies the work required to maintain coherence.

The system is no longer dominated by what it produces.

It is dominated by what it must reconcile.

This is the shift O’Laughlin identified without naming the source. Aggregation Theory ended because the architecture underneath inference had crossed into a regime where every additional interaction paid a coordination cost at every boundary it crossed.

Marginal cost returned when the product term became the dominant term.

The Trap

This is where the instinct fails.

More orchestration does not reduce coordination.
It introduces additional control surfaces.

More abstraction does not eliminate work.
It relocates it and increases the number of crossings.

More monitoring does not improve coherence.
It adds additional systems that must themselves be reconciled.

Each layer assumes it is reducing complexity.
Each layer introduces another independent dimension.

The system responds predictably.

Coordination cost increases faster than useful work.

Additive fixes accelerate the condition they attempt to correct.

There Is No Escape Inside the Model

If coordination cost grows multiplicatively, and energy is finite, then:

- optimization cannot resolve it
- scaling cannot resolve it
- abstraction cannot resolve it

All three operate within the same function.

None change its class.
None reduce the dimensionality of the system.

What You Are Seeing

The symptoms are already visible:

- systems that fail under coordination load, not compute load
- configurations that propagate without containment
- policy that fragments across domains
- recovery that becomes manual because the system cannot reconcile itself

The pipes work.

The system cannot govern what flows through them.

The Boundary

The system is not inefficient.

It is misclassified.

It is being treated as a linear system with optimizable overhead.

It is a non-linear system governed by a boundary condition.

That boundary is not reached gradually.

It is crossed.

When:

$k \cdot P \cdot M \cdot F \cdot A \cdot T \geq$ available capacity

the system ceases to produce output.

All available energy is consumed maintaining internal consistency.

Execution collapses.

O'Laughlin saw the end of Aggregation Theory.
Thompson saw the allocation constraint.
Huang described the energy ceiling.

Each was looking at a different face of the same condition.

Coordination grows as a product.
Compute does not.

As those constraints increase, the work required to maintain coherence increases with them, drawn from the same finite pool that must power the computation itself.

This is not an implementation problem.
It is not an optimization problem.
It is not a scaling problem.

It is a system boundary condition.

There is no path around it inside the current architecture.