You modeled compute.
You modeled storage.
You didn't model egress.

Cloud egress is the tax that accumulates invisibly — not from one expensive operation, but from thousands of small data movement events.

3 categories. Most teams only track 1.

01

Internet Egress

Cloud → User / External System

Everyone tracks this one

02

Cross-Region Egress

Region A → Region B, same provider

Most teams track this

03

Cross-Zone Egress

$0.01/GB each direction, every AZ hop

Almost nobody budgets this

Storage is cheap.
Moving data out of it isn't.

Analytics Queries

Scan full datasets on every execution — gigabytes from storage to compute each run

ML Training Pipelines

Pull training data from S3 into GPU instances — egress on every epoch

Data Pipeline Fan-Out

Copy data between storage tiers rather than transforming in place — 1TB becomes 4TB

The cost is not in storing the data. It is in every system that reads it.

>_ EGRESS MULTIPLIERS

The patterns that turn one egress event into ten.

Most egress cost analyses focus on individual data transfer events.
The real problem is architectural patterns that multiply egress.

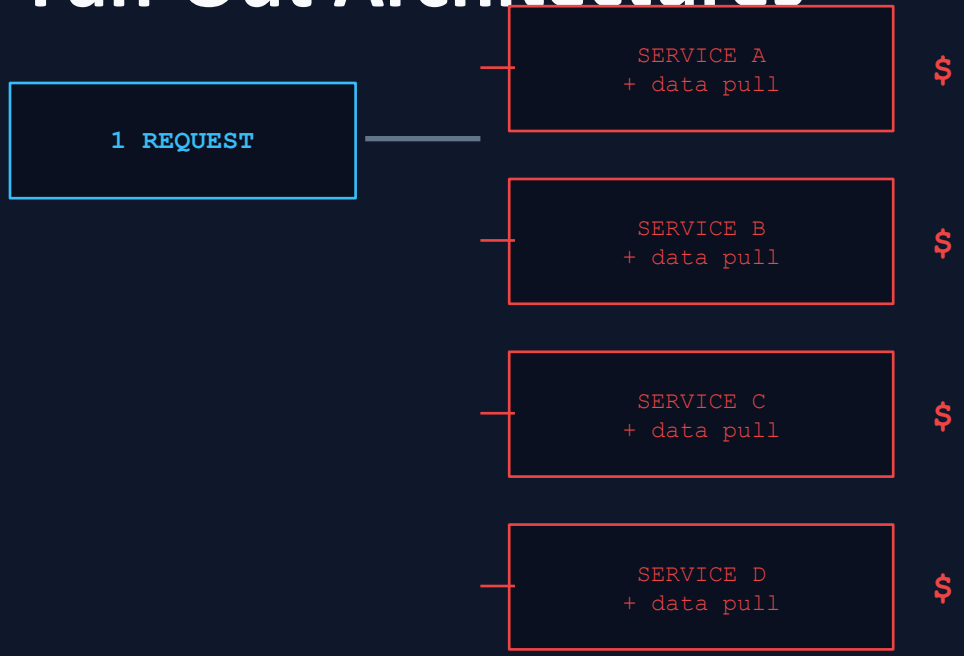
01 — Fan-Out Architectures

02 — Retry Storms

03 — Cross-Zone Chatter

04 — Data Duplication Pipelines

Fan-Out Architectures



1 request → N egress events

Fan-out is a correct design for scalability — it becomes an egress problem when the multiplier is never modeled against data transfer costs.

10 concurrent users = 100 egress events

02 Retry Storms

Failed requests retry with full payload. At scale, retry storms generate egress volume that exceeds the original traffic by multiples. Retry logic without exponential backoff turns brief degradation into a sustained egress event.

Same data. Transferred repeatedly.
Zero delivery.

03 Cross-Zone Chatter

Every AZ hop in a microservice call chain is metered at $0.01/GB each direction. A request chain across 5 services and 3 AZs = 5 potential cross-zone events. Zone-aware routing reduces this without changing the availability architecture.

`$0.01/GB × N hops × M requests/sec`

04 Data Duplication Pipelines

ETL that copies data between tiers — raw → processed → curated — generates egress at every stage. 1TB through 4 pipeline stages = 4TB transferred. Query in place with BigQuery/Athena instead.

`1TB pipeline × 4 stages = 4TB egress`

Egress rarely comes from a single path.

It comes from paths that multiply.

5 architecture levers. Not FinOps tweaks.

01

Collocate compute + data

Same region, same AZ. Zone-aware K8s scheduling.

02

Query in place

BigQuery, Athena, Redshift Spectrum — scan where data lives.

03

Cache aggressively

CDN for users. In-memory for inter-service. Every hit = zero egress.

04

Compress before transfer

zstd/Brotli: 60–80% reduction. Protobuf vs JSON: 3–10x smaller.

05

Audit the multipliers

Find fan-out, retry storms, cross-zone chatter before optimizing rates.

>_ ARCHITECT'S VERDICT

Egress is not a billing problem.
It's an architecture problem.

Model data movement as a first-class constraint at design time — not after the first invoice.

The patterns generating the largest bills are correct architectural decisions made without egress as a design input.

High availability across AZs, fan-out for scalability, retry logic for resilience — all correct. All unmodeled.

Model it like compute. Model it like storage. It is the same tax, arriving from a direction you didn't expect.

>_ READ THE FULL GUIDE + USE THE CALCULATOR

Cloud Egress Costs Explained

Why Your Architecture Is Paying a Tax You Never Modeled

>_ FULL GUIDE

rack2cloud.com/cloud-egress-costs-explained/

>_ TOOL: CLOUD EGRESS CALCULATOR

Model true data movement costs across AWS, Azure, and GCP.
Exposes the hidden tiered pricing models before the bill arrives.

rack2cloud.com/deterministic-tools-for-a-non-deterministic-cloud/