

TASK RUNNERS, DATA BINDINGS AND TWO KINDS OF GPU — CHALLENGE, SOLUTION, BENEFIT

rows 1 and 2 decide who may run and what it may touch; rows 3 and 4 decide whether data goes to the GPU or the GPU code goes to the data

Red dashed = the challenge. Rows 3 and 4 split GPU work by size; they don't compete.

CHALLENGE

1 · Task runner

Jobs with no owner

crons, webhooks and queue consumers
spread across tools; retries collide

2 · Data binding

Code that can reach anything

credentials and URLs in code, so any
function can read any store

3 · GPU as a place

GPUs are scarce and heavy

owning them means capacity planning
and long commitments

4 · GPU on the device

Small GPU work sent away

previews and 3D shipped to a cluster
the person's own GPU could run

SOLUTION

One runner per path

Worker cron and queues at the edge;
Xano tasks for the record; idempotency keys

Bindings declared in config

env.KV, env.DB, env.AI and one Xano
endpoint; nothing else is reachable

Rent a cluster

CoreWeave or Oracle OCI; NVIDIA-only
code (CUDA, PTX); data travels to it

Portable GPU bytecode

SPIR-V through wgpu on any GPU;
Rust is one source language of several

BENEFIT

Every job is accountable

an owner, a retry policy and a log
for each job

The config is the access list

an audit reads one file,
not the whole codebase

Training at scale

big models trained and served, but the
data leaves your boundary to get there

Small GPU work stays with the data

previews, real-time 3D, upscaling
run where the data already is