Video AI Cost Optimization - 25-Lever Checklist

Fora Soft Learn · AI for Video Engineering

The one idea - levers multiply, they don't add

```
Three independent 3x wins compound to 27x, not 9x. Pull EVERY lever that applies to your workload.
Two levers multiply only when they act on different parts of the cost (token count vs price per token).
```

The 25 levers, in five families

1-5 PROCESS LESS VIDEO (the biggest win for video)

1 Sample frames - don't process all 30 every second (1 fps = 30x fewer)
2 Downscale resolution before the model sees the frame
3 Crop to the region of interest with a cheap detector first
4 Gate the big model behind a cheap trigger (run on events, not a schedule)
5 Trim the system prompt; cap the output length (output tokens cost the most)

6-10 RUN A SMALLER, CHEAPER MODEL

6 Route the easy 90% to a cheaper tier (frontier is 8-30x the price)
7 Cascade: try the cheap model first, escalate only on low confidence
8 Swap a frontier API for a small open model where quality allows
9 Distil a smaller student model (e.g. Distil-Whisper, ~6x faster)
10 Quantize to INT8 / INT4 (~4x less memory, usually <1% accuracy)

11-15 SERVE THE MODEL MORE EFFICIENTLY (self-host)

11 Continuous batching - keep the GPU full (up to ~23x throughput)
12 Reuse the KV cache (PagedAttention)
13 Speculative decoding - 2-3x faster, mathematically identical output
14 A purpose-built runtime (vLLM / TensorRT-LLM / SGLang)
15 Right-size the GPU to the model's memory footprint

16-20 PAY LESS PER UNIT OF COMPUTE (mostly config, not code)

16 Batch API for non-real-time jobs - flat 50% off
17 Prompt caching - up to 90% off the repeated input prefix
18 Semantic answer caching + dedupe re-uploaded clips
19 Spot / preemptible GPUs - 60-90% off (interruptible work only)
20 Reserved / committed capacity - 20-70% off the steady baseline

21-25 ARCHITECT & GOVERN FOR COST

21 Serverless scale-to-zero - pay per second, idle costs nothing
22 Right provider (neocloud vs hyperscaler); kill egress/storage waste
23 Hybrid edge + cloud - constant work on-device, hard work in the cloud
24 Precompute derived artifacts once (transcript, embeddings, index)
25 Instrument per-feature cost budgets; alert when one drifts

Worked example - a 50,000-hour/month moderation pipeline

```
Naive:  1.08M tok/hr x $2.50/1M = $2.70/hr x 50,000 = $135,000 / month
+ process less video (low-res + sampling, ~1/3 tokens)  -> $45,000   (/3)
+ route the easy 90% to a Flash tier (blended $0.52/1M)  -> $9,360    (/4.8)
+ run the offline job on the batch API (-50%)            -> $4,680    (/2)
Three independent levers -> 29x cheaper, with no change the user sees.
```

Before you optimize, ask:

- [ ] Have we instrumented cost per feature BEFORE optimizing - do we know where the money actually goes?
- [ ] Are we processing less video first (frame sampling, low resolution, ROI crop, event-gating)?
- [ ] Is routine work on a cheaper model tier, with the flagship kept for the hard minority?
- [ ] Is every non-real-time job on the batch API, with shared prompts cached (up to 90% off)?
- [ ] Does the pricing mode match the load (spot = interruptible, reserved = baseline, serverless = spiky)?
- [ ] For self-hosted work, are we on an efficient runtime with continuous batching and a right-sized GPU?
- [ ] Are the stacked levers INDEPENDENT, and did we verify the total against a real measurement?