BÁO CÁO ĐO TẢI · 09/09/2026

Trần chịu tải chandra-ocr-2-dla

Ba đợt đo trên 1×H200: sweep tham số vLLM tìm trần thật, truy nút thắt Spark bằng quan sát trực tiếp GPU, và sweep VRAM tìm ngưỡng bộ nhớ tối thiểu.

1×H200 141GB (GPU1) 36 điểm đo 852 trang qua Spark 100% output đã validate

01 Ba con số

TRẦN GPU

8.02

trang/giây — 28,868 trang/giờ

CCU CÂN BẰNG

64

6.24 trang/s, p95 13.4s

VRAM TỐI THIỂU

33

GiB — thay vì 113 GiB

SPARK (10K TRANG)

3.97

trang/giây — 64% trần

Cấu hình khuyến nghị:

```
--max-num-seqs 128 --max-num-batched-tokens 65536 --gpu-memory-utilization 0.28
```

> **Cả ba con số đều khác báo cáo đầu.** Bản đầu kết luận trần 2.59 trang/s ở 16 CCU — đó là trần của *một cấu hình cụ thể*, không phải của phần cứng. Sweep tham số server cho thấy trần thật cao **gấp 3.2 lần**, và sweep VRAM cho thấy chỉ cần **29% bộ nhớ** đang cấp.

02 Đợt 1 — Sweep tham số vLLM

Cấu hình gốc `max-num-seqs=16` chặn cứng ở 16 request đồng thời, trong khi vLLM cấp phát 109 GiB KV cache đủ cho concurrency 98x. Log lúc chạy đầy tải: `GPU KV cache usage: 2.0%`, `Waiting: 0` — **98% KV cache bị bỏ không.**

SEQS:BATCHED	CONC 16	CONC 32	CONC 64	CONC 96	CONC 128
16:8k (gốc)	2.496	2.531	2.575	–	–
32:16k	2.477	3.859	3.841	–	–
32:32k	2.500	3.895	3.858	–	–
64:16k	2.486	3.855	**5.559**	–	–
64:32k	2.474	3.860	**5.638**	–	–
128:32k	–	–	5.458	6.552	**7.351**
128:64k	–	–	6.149	7.318	**8.019**
256:64k	–	–	6.244	7.265	**8.007**

Trang/giây, chỉ đếm output đã validate. Hàng đầu là cấu hình gốc — nằm bẹp ở 2.5 dù client tăng gấp 4.

Ba quy luật

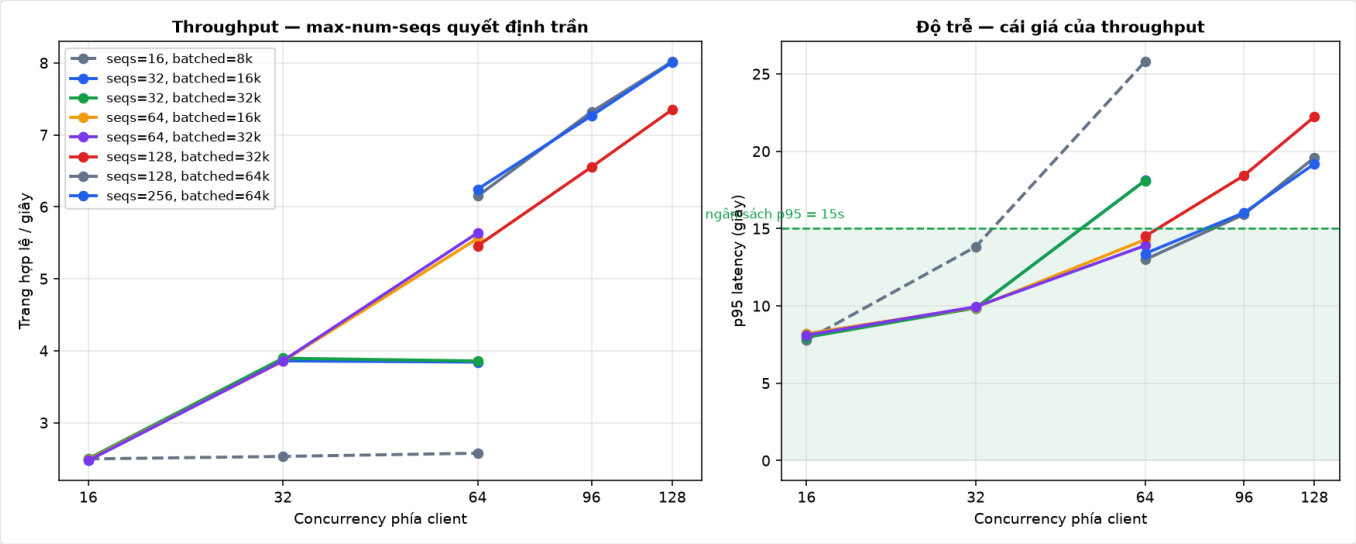
1. **Cấu hình server vô dụng nếu client không gửi đủ tải**
   Cột conc 16: mọi cấu hình đều ~2.5 trang/s. Nâng `max-num-seqs` mà client vẫn gửi 16 luồng thì không được gì.

2. **max-num-seqs bão hoà ở 128**
   Nâng lên 256 chỉ hơn 1.5% — trong sai số. Vượt quá concurrency thực tế còn hơi hại.

3. **max-num-batched-tokens chỉ quan trọng ở tải cao**

✓ max-num-batched-tokens chỉ quan trọng ở tải cao

Ở conc ≤ 32, 16k hay 32k không khác gì. Ở conc 64, `128:64k` thắng `64:32k` cả throughput (6.149 vs 5.638) lẫn latency (12.99s vs 13.90s).



Đường xám đứt (cấu hình gốc) vừa chậm nhất vừa trễ nhất — ở conc 64 nó là đường duy nhất vọt lên 25.8s.

03 Đợt 2 — Nút thắt Spark

Trên cấu hình tối ưu, Spark chỉ đạt 1.84 trang/s = 30% trần. Tôi đưa ra hai giả thuyết và **cả hai đều sai:**

✗ "Cụm Spark thiếu core"

Nâng 12 → 24 core: 1.827 vs 1.837 trang/s — không đổi gì.

✗ "RDD bị tính lại hai lần"

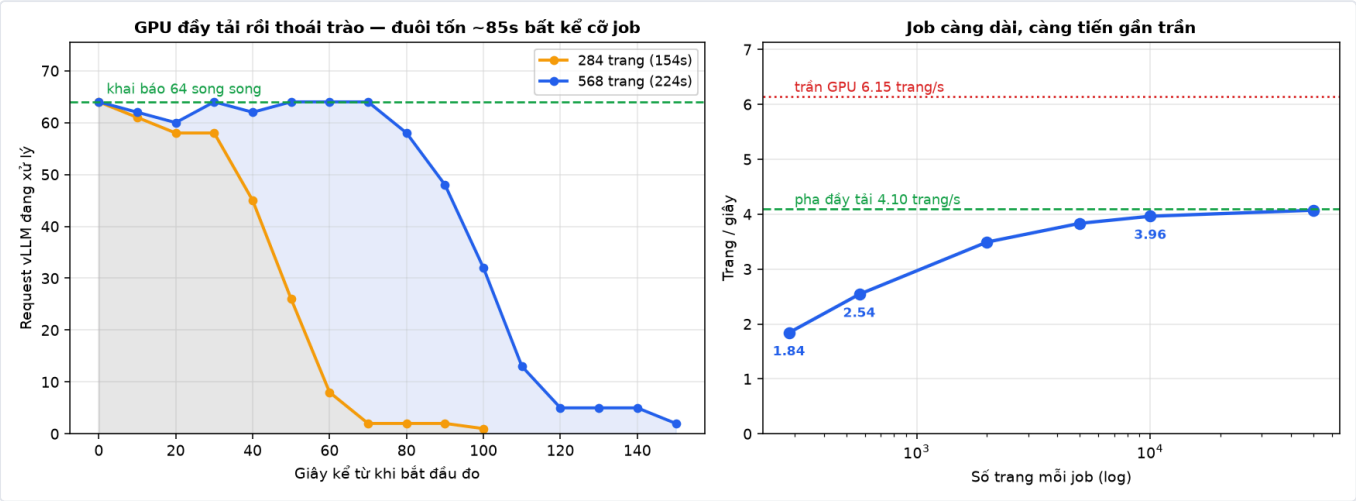
Code có hai `collect()` trên RDD chứa ~800MB PNG. Gộp thành một action: 1.839 — vẫn không đổi.

Phải lấy mẫu `Running: N reqs` từ log vLLM trong lúc Spark chạy mới ra:

CỠ JOB	CHUỖI RUNNING (MỖI 10S)	TB	WAITING
284 trang	64 61 58 58 45 26 8 2 2 2 1	29.7	0
568 trang	**64 62 60 64 62 64 64 64 58 48 32 13 5 5 5 2**	**42.0**	**0**

> **Spark gửi đủ 64 request đồng thời** — GPU không đói việc, `Waiting: 0` suốt. Nút thắt là **hiệu ứng đuôi**: khi gần hết trang, concurrency tụt dần về 1 và GPU chạy non tải.

Nhân đôi dữ liệu để kiểm chứng: 568 trang xong trong 224s (chỉ tăng 45% dù dữ liệu gấp đôi), goodput tăng 38%. Pha đầy tải dài gấp đôi, đuôi vẫn tốn đúng một lần.



Trái: cả hai job đạt 64 song song rồi thoái trào; job dài giữ đầy tải lâu hơn. Phải: mô hình `wall = trang/4.10 + 85s` khớp cả hai điểm đo.

SỐ TRANG/JOB	THỜI GIAN	TRANG/S	% TRẦN
284	2.6 phút	1.84	30%
568	3.7 phút	2.54	41%
2,000	9.5 phút	3.49	57%
10,000	**42.1 phút**	**3.96**	**64%**
50,000	204.7 phút	4.07	66%

Con số 30% ở bộ mẫu 284 trang là **ảo giác do job quá ngắn.** Với khối lượng thật, Spark đạt 62–66% trần mà không cần sửa gì.

04 Đợt 3 — Ngưỡng VRAM tối thiểu

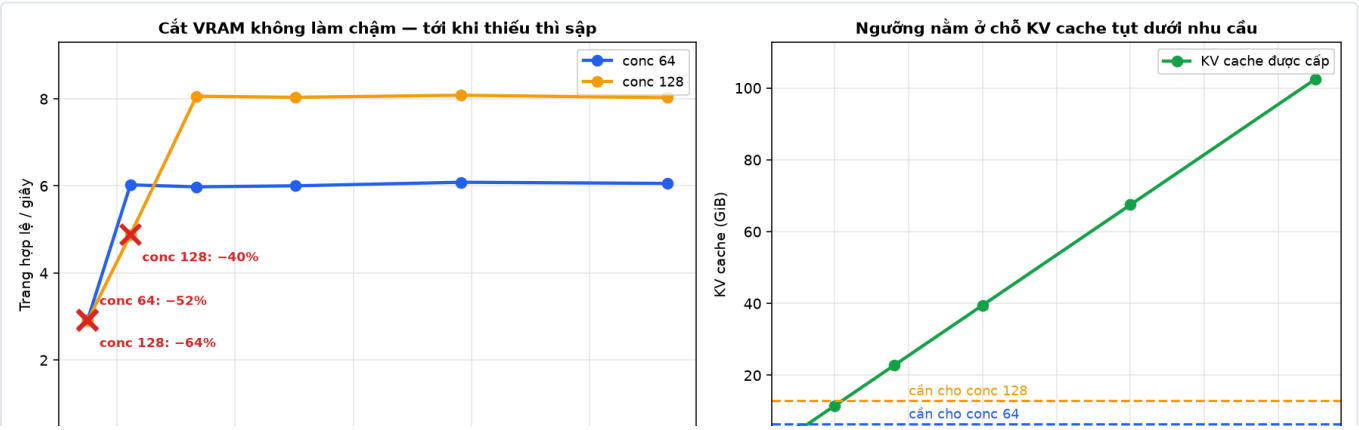
Giảm dần `gpu-memory-utilization` từ 0.85, đo lại ở conc 64 và 128 mỗi mức.

UTIL	VRAM (MIB)	KV CACHE (GIB)	CONC 64	CONC 128
0.85	115,841	102.4	6.058 ✓	8.027 ✓
0.60	80,051	67.5	6.084 ✓	8.084 ✓
0.40	51,407	39.5	6.003 ✓	8.036 ✓
0.28	34,231	22.7	5.979 ✓	8.060 ✓
0.20	22,781	11.6	6.025 ✓	4.878 -39%
0.15	15,389	4.6	2.930 -52%	2.908 -64%
0.12	vLLM không khởi động được — KV cache ~0.4 GiB, không đủ chỗ			

> **Cắt VRAM không làm chậm — cho tới khi chạm ngưỡng thì sập hẳn.** Bốn mức từ 0.85 xuống 0.28 cho kết quả không phân biệt được về thống kê: độ lệch chuẩn 0.3% (conc 128) và 0.7% (conc 64). Cùng cấu hình đo hai lần khác nhau cho 8.019 và 8.027 — chênh 0.1%.

> **Ngưỡng phụ thuộc concurrency, không phải một con số duy nhất.** Ở util=0.20 (KV 11.6 GiB): conc 64 vẫn chạy bình thường, nhưng conc 128 tụt 39% vì cần ~12.9 GiB. Xuống util=0.15 (KV 4.6 GiB) thì cả hai đều sập. Ở util=0.12 vLLM không khởi động được.

Lý do KV cache dư không giúp gì: nó chỉ là chỗ chứa. Đủ chỗ cho 128 request thì xong; cấp thêm chỗ cho 1000 request không làm GPU tính nhanh hơn. Tốc độ do compute quyết định, không phải dung lượng. Nhưng *thiếu* chỗ thì vLLM phải preempt — đẩy request ra rồi tính lại — và tốc độ sập.





Trái: cả hai đường phẳng lì rồi rơi thẳng đứng — không có vùng suy giảm dần. Phải: ngưỡng nằm đúng chỗ đường KV cache cắt qua mức nhu cầu.

Chọn util theo mức tải

A **Cần conc 128 → `util 0.28` = 33 GiB** `an toàn`

Giữ nguyên tốc độ ở cả hai mức tải, còn biên 76% trên nhu cầu. Giải phóng **80 GiB**.

B **Chỉ cần conc 64 → `util 0.20` = 22 GiB** `sát ngưỡng`

Vẫn đủ tốc độ nhưng biên chỉ 78% trên nhu cầu conc 64 — nếu trang dài hơn bộ mẫu thì có thể preempt. Giải phóng 91 GiB.

05 Chất lượng output

Mọi con số throughput chỉ đếm trang có output **đã validate** — không tính trang bị cắt (`finish_reason=length`), trả plain-text, hay sai nhãn. Toàn bộ 36 điểm đo đạt 100% hợp lệ; Spark đạt 567/568 trang.

Ba lỗi đã sửa trong quá trình đo:

- **Nhãn biến thể** — `Page-Header` , `Page Header` → chuẩn hoá về `Header` .
- **Plain-text** — model bỏ qua yêu cầu HTML, `parse_blocks()` trả mảng rỗng và trang **biến mất im lặng** khỏi bundle. 11/284 trang rơi vào ca này.
- **Lặp vô hạn** — biểu mẫu có dòng chấm điền tay khiến model sinh `. . . .` vô tận, chạm 16384 token sau 111s.

> **Không bật frequency_penalty mặc định.** A/B trên 28 trang: bật mặc định làm hỏng 5 trang vốn tốt (28/28 → 23/28), gây `unclosed_div` . Chỉ bật ở lần retry sau khi trang đã chạm trần token.

06 Khuyến nghị

1 **Đổi cấu hình vLLM** `tiết kiệm ~70% VRAM`

1 **Đổi cấu hình vLLM** +146% tốc độ, -70% VRAM

`--max-num-seqs 128 --max-num-batched-tokens 65536 --gpu-memory-utilization 0.28`

2 **Spark chạy tasks×threads = 64** đã đo

`--tasks 8 --threads 8` , cụm 24 core. Song song client phải khớp `max-num-seqs` .

3 **Gộp job thay vì chạy nhiều job nhỏ** đã đo

Đuôi tốn 85s mỗi job bất kể cỡ. 10.000 trang chạy một lần nhanh gấp **2.2 lần** so với chia 35 job nhỏ.

4 **80 GiB giải phóng dùng làm gì** cần bàn

Đủ chỗ cho một replica chandra thứ hai trên cùng GPU1, hoặc dời một service từ GPU0 sang.

5 **Rút ngắn đuôi bằng partition mịn hơn** chưa đo

Tăng số partition để trang cuối phân bố đều hơn. Có thể thu hồi phần lớn 85s.

07 Chạy lại

```
# Sweep tham số server + VRAM (tự khôi phục cấu hình gốc khi xong)
cd idp-bench
python sweep_server.py --configs 128:65536:0.28 128:65536:0.20 \
    --concurrency 64 128 --duration 60 --out results/sweep_vram.json

# Spark ETL có đo concurrency GPU
bash diagnose_spark.sh

# Nếu sweep bị ngắt giữa chừng:
bash chandra-dla-deploy/restore.sh
```

1×H200 (GPU1) · vLLM 0.29.1 · 14 PDF tiếng Việt (284 trang) và bản nhân đôi (568 trang), 200 DPI
36 điểm đo, tất cả 100% output đã validate. Cấu hình gốc được khôi phục sau mọi phép đo.