DynaAI-engine の負荷特性評価

—— MoE / dense モデルと配置構成が同時利用者数に与える影響 ——

永山 和樹　　林 孝太朗　　大和田 昭邦

株式会社 DynaxT 新技術商品開発部　　株式会社 DynaxT　　株式会社 DynaxT

2026 年 7 月 27 日

概要

目的：社内 AI 基盤 DynaAI-engine を業務システムへ組み込むにあたり、「何人までの同時利用に耐えられるのか」「GPU を増やせばどれだけ伸びるのか」「どのモデルを選ぶべきか」の 3 点を実測によって明らかにする。

方法：OpenAI 互換エンドポイントに対し、負荷試験ツール aiperf を用いて同時実行数を倍々に増やしながら各段で定常負荷をかけ、スループットが飽和するまで自動でスイープした。スループットとレイテンシ（TTFT・トークン間レイテンシ）の双方を測定したうえで、レイテンシ制約を満たす最大の同時実行数を**実用飽和点**として同定した。対象は MoE 2 種・dense 1 種のモデルである。

結論：第一に、飽和点の判定にはスループット曲線よりも TTFT のほうが鋭敏である。第二に、スループットを規定するのは総パラメータ数でも MoE か dense かの別でもなく、**活性パラメータ数**であった。第三に、GPU 1 枚に収まるモデルをデータ並列で並べる構成は、レプリカ本数どおりに素直にスケールする。そして最も重要な点として、**本試験で観測された飽和点は GPU の計算能力ではなく、推論エンジン側で設定したレプリカあたり同時受け入れ数の上限に規定されていた**。したがって本稿が示す収容人数は**現行設定における値**であってハードウェアの上限ではなく、容量を増やすにはまずこの設定値を見直すべきである。

目次

1 経緯

社内 AI 基盤 DynaAI-engine を業務システムへ組み込むにあたり、「**何人まで同時に使えるのか**」「**GPU を増やせばどれだけ伸びるのか**」「**どのモデルを選ぶべきか**」という 3 つの問いに答える必要が生じた。そのため、実測を行った。

評価軸は**スループット**（単位時間あたりの処理量）と**レイテンシ**（利用者の体感）の 2 つであり、両者のトレードオフが崩れる点＝**飽和点**を同定することを目的とする。

なお本稿はモデルの**回答品質**を評価するものではない。

2 試験環境

試験に用いたハードウェア・ソフトウェア構成を表 1 に示す。AI サーバは NVIDIA A6000 を 10 枚搭載し、そのうち 2 / 4 / 8 枚を推論エンジンに割り当てて比較した。カード間の接続は**一様ではない**。GPU 1–2 / 3–4 / 6–7 / 8–9 の 4 組が **NVLink** で対向接続され、それ以外のカード間（ペアをまたぐ経路、および GPU 5・GPU 10）は **PCIe** 経由となる（図 1）。すなわち **2 枚までは NVLink の帯域を使えるが、3 枚以上を 1 つのモデルで束ねる場合は必ず PCIe を跨ぐ**という非対称なトポロジである。

2.1 並列方式

モデルによって配置方式が異なる。この違いは結果の解釈を大きく左右するため、先に整理しておく。

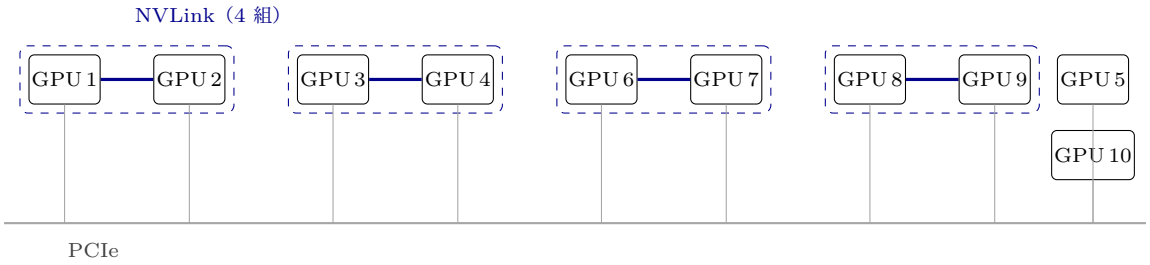

図 1　GPU 間の接続トポロジ。青の実線が NVLink、灰の線が PCIe。GPU 1–2 / 3–4 / 6–7 / 8–9 の 4 組のみ NVLink で対向接続されており、ペアをまたぐ通信は PCIe を経由する。

表 1　試験環境

区分	項目	内容
AI サーバ (HW)	CPU	Xeon Gold 6326 ×2（32C/64T）
	RAM	1024 GB
	GPU	NVIDIA A6000（VRAM 48 GB）×10 枚
	GPU 割当	2 / 4 / 8 枚
	GPU 間接続	NVLink（1–2 / 3–4 / 6–7 / 8–9 の 4 組）、他は PCIe
AI サーバ (SW)	推論エンジン	OSS の LLM 推論サーバ（試験期間中はバージョン固定）
	小モデル	MoE、総 26B / 活性 4B、FP8
	中モデル	dense、31B、FP8
	大モデル	MoE、総 122B / 活性 10B、FP8
並列方式	小・中モデル	データ並列（1 GPU = 1 レプリカ、ロードバランサで分散）
	大モデル	単一インスタンスで TP 4 × PP 2（GPU 8 枚）
エンドポイント	種別	OpenAI 互換 `/v1/chat/completions`（streaming）
負荷クライアント	ツール	aiperf（ai-dynamo/aiperf）
	実行形態	Docker（python:3.11-slim ベース）
	トークナイザ	各モデルの HuggingFace tokenizer

- **小モデル・中モデル：データ並列.** GPU 1 枚につき推論エンジンを 1 プロセス（テンソル並列なし）起動して独立したレプリカとし、その手前にロードバランサを置いて振り分ける。小モデル (FP8 [9]) は約 26 GB、中モデル (FP8) は約 31 GB でいずれも A6000 1 枚（48 GB）に収まる。**レプリカ内でカードを跨ぐ通信は発生しない**ため、小モデル・中モデルの測定では GPU 間トポロジ（図 1）は性能に影響しない。「×N 枚」とはすなわち**レプリカ N 本**を意味する。
- **大モデル：単一インスタンスの TP 4 × PP 2.** FP8 でも約 122 GB あり 1 枚に収まらないため、テンソル並列 [6] 4 × パイプライン並列 [7] 2 で GPU 8 枚を 1 レプリカとして束ねている。**本試験で唯一、カード間の集団通信が発生する構成**である。TP グループは 4 枚から成るので NVLink ペア（2 枚）には収まらず、**必ず PCIe を跨ぐ**。

とくに重要な設定値を表 2 に抜き出す。**小モデル・中モデルのレプリカあたり同時受け入れ数の上限 8**（1 レプリカが同時に処理を始められるリクエスト数）は、第 5.2 節で述べるとおり本試験の飽和点をほぼ直接に決めている。

表 2　結果の解釈に影響する主な推論エンジン設定

設定	小モデル / 中モデル	大モデル
テンソル並列数	1	4
パイプライン並列数	1	2
同時受け入れ数の上限	**8**（レプリカあたり）	256
最大コンテキスト長	8192	262144
1 ステップあたり最大バッチトークン数	（既定）	**4096**
GPU メモリ使用率	0.90	0.90
KV キャッシュのデータ型	（既定）	FP8
プレフィックスキャッシュ	有効	有効
投機的デコード	**有効**（draft モデル、3 トークン先読み）	なし

3　試験方法

3.1　負荷パターン

負荷生成には NVIDIA ai-dynamo の aiperf [10] を用いた。実業務での RAG（Retrieval-Augmented Generation）[1] 利用を想定し、入力長 2048 トークン・出力長 512 トークンを目標値として aiperf にプロンプトを自動生成させた。試験パラメータを表 3 に示す。

表 3　試験パラメータ

項目	値	備考
入力トークン長 (ISL)	平均 2048	RAG のコンテキスト長を想定
出力トークン長 (OSL)	平均 512	標準的な回答長を想定
測定時間	300 秒	各同時実行数ごと
ウォームアップ	30 秒	測定対象外
同時実行数 c	1, 2, 4, 8, ..., 256	倍々に増加・上限 256
飽和判定閾値	前段比 +5% 未満	スイープ停止の判定に用いる
飽和判定の連続回数	1 回	1 度でも伸びなければ停止する
エンドポイント種別	chat (streaming)	TTFT / ITL 測定のため必須

3.2　自動飽和スイープ

同時実行数を人手で選ぶと飽和点を跨いだり手前で止めたりしやすいため、スイープを自動化した（図 2）。c を 1 から倍々に増やし、各段で 300 秒の定常負荷をかけたのち `request_throughput` を前段と比較する。前段比が +5% 未満であれば「これ以上並列を増やしてもスループットは伸びない」と判断してスイープを停止する。上限 256 は暴走防止の安全弁である。

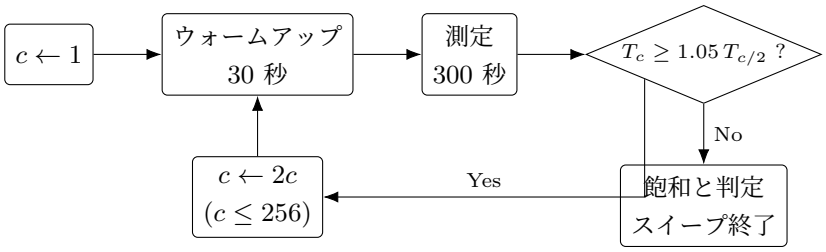

図 2　自動飽和スイープの制御フロー（T_c は同時実行数 c での `request_throughput`）

3.3　評価指標

aiperf が出力する指標のうち、本稿では表 4 の 5 つを用いる。ストリーミング応答の体感は TTFT（待たされ感）と ITL（文字の流れる速さ）の 2 段構えで決まるため、両者を分けて評価する。

表 4　評価指標

指標	単位	意味
`request_throughput`	req/s	システム全体で 1 秒あたり処理できたリクエスト数
`output_token_throughput`	tok/s	システム全体の生成トークン数（集約スループット）
`request_latency`	ms	リクエスト受付から応答完了までの時間
`time_to_first_token` (TTFT)	ms	最初の 1 トークンが返るまでの時間
`inter_token_latency` (ITL)	ms	トークンとトークンの間隔。1000/ITL が利用者 1 人あたりの体感生成速度 [tok/s]

3.4　実用飽和点の定義

スクリプトによる自動停止（前段比 +5% ルール）は「スループットがこれ以上伸びない点」を検出するが、その時点ではすでにレイテンシが実用に耐えない水準まで悪化している場合がある。そこで本稿では、レイテンシ制約を加えた**実用飽和点** c_{sat} を式 (1) のように定義する。

$$c_{\mathrm{sat}} = \max\Big\{\, c \;\Big|\; \mathrm{TTFT}_{\mathrm{avg}}(c) \leq 4000\,\mathrm{ms} \;\wedge\; \mathrm{ITL}_{\mathrm{avg}}(c) \leq 50\,\mathrm{ms} \Big\} \tag{1}$$

TTFT 4 秒は、後述するとおり全構成において TTFT が急峻に立ち上がる直前の水準であり、キュー待ちが顕在化し始める境界に相当する。ITL 50 ms は利用者 1 人あたり 20 tok/s に相当し、日本語の読み上げ速度を上回る最低限の水準である。

3.5　測定手法の妥当性

本稿の測定手法は、推論サービングのベンチマークで確立した作法に沿っている。対応関係を表 5 に示す。

一方で、本稿の手法には**原理的な限界が 2 つ**ある。いずれも第 5.9 節に制約として再掲するが、手法の位置づけに直結するため先に述べる。

表 5　本稿の測定手法と、その根拠となる先行研究

本稿の手法	根拠	何が保証されるか
レイテンシ制約下の最大スループットを性能指標とする（式 (1)）	[11]	業界標準ベンチマークの Server シナリオが同じ定義を採る
同上	[12]	レイテンシ SLO を満たすリクエストのみを数える goodput の考え方と一致
TTFT と ITL を分けて評価する	[3]	ストリーミング応答の体感をこの 2 指標で捉えるのが標準
飽和後の並列増加が待ち時間に転化する（第 5.1 節）	[14]	c = スループット × レイテンシという関係の形式的根拠
ウォームアップ 30 秒を分離し、定常状態の 300 秒で測る	[15]	初期の非定常区間を測定に含めないこと
平均だけでなく p99 も併記する	[16, 17]	分布・テールを報告すべきこと

■(1) 閉ループ負荷生成であること． aiperf に同時実行数 c を指定して負荷をかける方式は、**常に c 本のリクエストを飛ばし続ける「閉ループ (closed-loop)」**の負荷生成である。実際の利用者トラフィックは到着率が外から決まる「開ループ (open-loop)」に近く、両者は応答時間の分布が定性的に異なることが知られている [13]。閉ループでは飽和後もリクエスト数が増えないため系が破綻せず、待ち時間が伸びるだけで済むが、開ループでは到着率が処理能力を超えた時点で待ち行列が発散する。したがって**本稿の飽和点は「同時にこれだけ処理できる」という容量の指標であって、「毎秒これだけの到着に耐えられる」という指標ではない**。第 5.4 節でアイドル率 ρ を掛けて人数へ換算しているのは、この差を粗く埋める近似にすぎない。

■(2) 飽和判定に平均を用いていること． 式 (1) は TTFT と ITL の**平均**に制約を課している。標準的なベンチマークはテール（p99 など）で制約を課すため [11, 17]、平均による判定は緩い側に倒れる。そこで p99 に基づく飽和点も併せて定義し、

$$c_{\mathrm{sat}}^{p99} = \max\left\{ c \;\middle|\; \mathrm{TTFT}_{p99}(c) \le 4000\,\mathrm{ms} \;\wedge\; \mathrm{ITL}_{p99}(c) \le 50\,\mathrm{ms} \right\} \tag{2}$$

第 5.3 節で両者を比較する。

4　結果

4.1　全体サマリ

7 構成それぞれについて、自動停止した同時実行数、実用飽和点、およびそこでの各指標を表 6 に示す。

表 6　構成別サマリ（実用飽和点は式 (1) による）

構成	自動停止時の c	実用飽和点における値 c_{sat}	req/s	tok/s	tok/s/GPU	遅延 [s]	TTFT[ms]	ITL[ms]	最大 req/s
小モデル ×2	16	8	0.84	428	214	9.40	539.4	17.3	0.85
小モデル ×4	128	32	3.00	1532	383	10.48	2010.6	16.6	3.45
小モデル ×8	128	64	5.68	2907	363	11.05	2586.3	16.6	5.88
中モデル ×2	32	8	0.37	187	93	21.15	2646.2	36.5	0.47
中モデル ×4	128	16	0.70	355	89	21.97	3607.6	36.3	0.94
中モデル ×8	128	32	1.39	705	88	22.25	3415.5	37.3	1.86
大モデル ×8	256	16	0.87	446	56	18.35	2191.9	31.6	1.31

4.2　スループット

集約スループット（req/s）の同時実行数依存性を図 3 に示す。いずれの構成でも低負荷域では c に比例して伸び、ある点から頭打ちになる典型的な飽和曲線を描く。

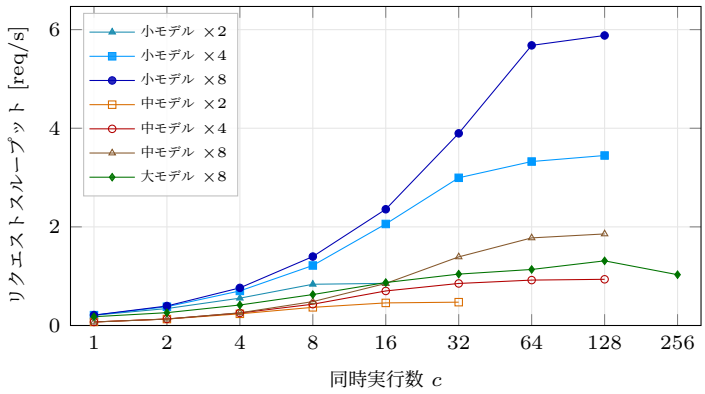

図 3　リクエストスループット vs 同時実行数

最大スループットは 小モデル ×8 枚の **5.88 req/s**（3,011 tok/s）であった。同じ GPU 8 枚で比べると 小モデル は 中モデル の**約 3.2 倍**、大モデル の**約 4.5 倍**のリクエストを捌く。

4.3　レイテンシ

リクエスト全体の平均レイテンシを図 4 に、TTFT 平均を図 5 に示す。

TTFT の挙動は明瞭である。飽和点までは TTFT はほぼ横ばいで低く保たれ、飽和点を超えると**1 桁単位で跳ね上がる**。たとえば 小モデル ×4 枚では $c = 32$ で 2,011 ms だったものが $c = 64$ で 9,962 ms、$c = 128$ で 25,859 ms へと悪化する。これは飽和後の並列増加分がすべてキュー待ち時間に転化するためであり、飽和点の同定において TTFT はスループット曲線よりも鋭敏な指標となる。

4.4　トークン間レイテンシ（ITL）

ITL 平均を図 6 に示す。ITL は利用者 1 人あたりの体感生成速度を直接規定するため、チャット用途では TTFT より支配的になりやすい。

小モデル は $c = 128$ でも ITL 平均 16.7 ms（60 tok/s）を維持し、最も体感が安定していた。中モデル は 23–44 ms と全域で許容内に収まるが、低負荷時点ですでに 小モデル の約 2.8 倍遅い。大

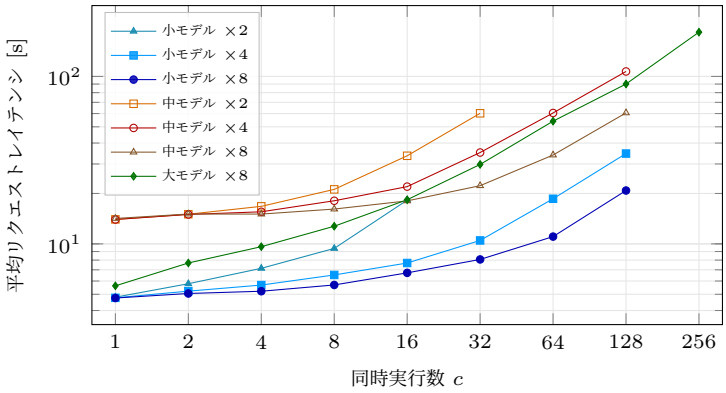

図 4　平均レイテンシ vs 同時実行数（対数軸）

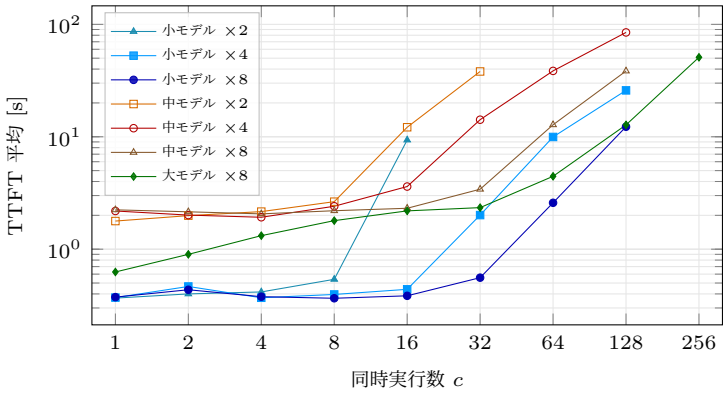

図 5　TTFT 平均 vs 同時実行数（対数軸）

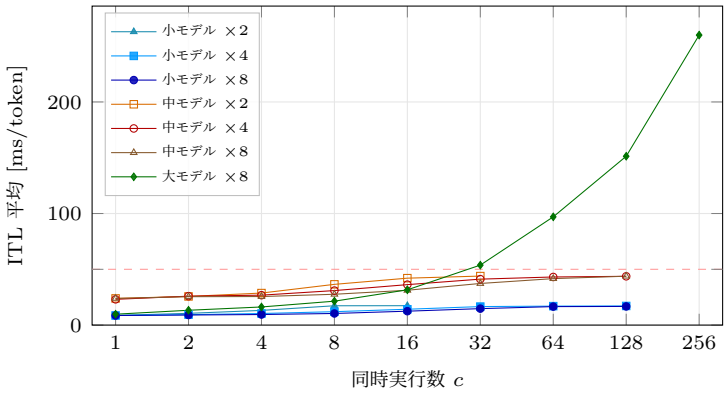

図 6　ITL 平均 vs 同時実行数（赤破線は式 (1) の許容上限 50 ms）

モデル は $c = 32$ で 53.8 ms と閾値を超え、$c = 256$ では 259.9 ms（3.8 tok/s）まで悪化した。本試験で ITL が実用飽和点の律速要因となったのは 大モデル のみである。

4.5　レプリカ本数に対するスケーラビリティ

小モデル・中モデルはデータ並列であるから、「×N 枚」はレプリカ N 本を意味する。実用飽和点における集約スループットのレプリカ本数依存性を表 7 に示す。

中モデル は ×1.90、×1.99 とほぼ完全な線形スケールを示した。レプリカを 2 倍にすれば集約

表 7　レプリカ本数に対するスケーリング（実用飽和点の出力 tok/s）

モデル	×2 枚	×4 枚	×8 枚	2 → 4 倍率	4 → 8 倍率
小モデル (MoE)	428	1,532	2,907	×3.58	×1.90
中モデル (dense)	187	355	705	×1.90	×1.99
大モデル (MoE)	—	—	446	—	—

スループットも 2 倍になるという、データ並列に期待される挙動そのものである。

一方 小モデル は 4 → 8 枚は ×1.90 で同様だが、2 → 4 枚が ×3.58 と線形を大きく超えている。データ並列で超線形が起きることは原理的にありえないため、これは **×2 構成の測定値が低く出ていること**を意味する。実際、GPU 1 枚あたりスループットは ×2 で 214 tok/s、×4 で 383 tok/s、×8 で 363 tok/s であり、小モデル ×2 のみが約 6 割の効率にとどまる。この点は第 5.2 節および第 5.6 節で検討する。

なお 大モデル は 8 枚構成のみの測定である（データ並列ではなく TP 4 × PP 2 の単一レプリカであり、そもそも本節の比較対象にならない）。

5　考察

5.1　飽和点の判定には TTFT を用いるべきである

スループット曲線の頭打ちのみで飽和を判定すると、実用上は過大な同時実行数を採用してしまう。たとえば 小モデル ×4 枚のスループットは $c = 32$ の 3.00 req/s から $c = 128$ の 3.45 req/s まで名目上は 15% 伸びるが、同区間で TTFT 平均は 2.0 秒から 25.9 秒へと **13 倍**悪化している。すなわち飽和後の「伸び」は、利用者を待たせることで得た見かけの数字にすぎない。

この挙動は連続バッチング方式の推論サーバに共通するものであり、スループットとレイテンシのトレードオフとして知られる [2, 3]。TTFT が飽和点で急峻に折れ曲がる（図 5）のに対し、スループットは緩やかに漸近するため、**TTFT のほうが飽和点の指標として鋭敏**である。本稿の式 (1) はこの観察に基づく。

5.2　飽和点は同時受け入れ数の上限が決めている

小モデル・中モデルの各レプリカは、同時に受け入れるリクエスト数の上限を 8 として起動されている。すなわち**レプリカ 1 本が同時に処理できるのは 8 リクエストまで**であり、N 本のレプリカからなる構成の受け入れ容量は $8N$ である。これを実測の飽和点と並べたものが表 8 である。

小モデル の 4 枚・8 枚構成は**実用飽和点が容量 $8N$ とぴたりと一致**し、TTFT が 1 桁跳ね上がるのはその倍の $16N$ である。容量を超えた分はレプリカに入れず待たされるのだから、当然の対応関係といえる。

中モデル は 2 枚・4 枚・8 枚のいずれも $c_{\mathrm{sat}}/N = 4$、すなわち容量の半分にとどまる。これは 1 トークンあたりの計算が重く、8 本を同時に走らせると TTFT 4 秒の制約を先に超えてしまうためである。**3 構成で値が完全に揃っている**ことは、レプリカ 1 本あたりの挙動が本数によらないというデータ並列の性質を裏づける。

表 8　設定上の受け入れ容量と実測の飽和点

構成	容量 $8N$	実用飽和点 c_{sat}	c_{sat}/N	TTFT が急騰する c
小モデル ×2	16	8	4	16
小モデル ×4	32	32	8	64
小モデル ×8	64	64	8	128
中モデル ×2	16	8	4	16
中モデル ×4	32	16	4	32
中モデル ×8	64	32	4	64

例外は**小モデル ×2 だけ**である。容量 16 に対し c_{sat} は 8（レプリカあたり 4）で、同じモデルの 4 枚・8 枚構成が 8 まで使えているのと食い違う。また TTFT が急騰するのも容量と同じ $c = 16$ であり、本来キュー待ちが生じないはずの水準ですでに 9,357 ms に達している。第 5.6 節でこの点を検討する。

この対応関係は、本稿の数値の意味を大きく限定する。**測定された飽和点は GPU の計算能力の限界ではなく、設定値の限界である。**実用飽和点における ITL は 小モデル で 16.6 ms（60 tok/s/人）にとどまっており、GPU がまだ余力を残していることを示唆する。この上限を引き上げれば、VRAM（KV キャッシュ）とレイテンシが許す範囲でスループットはさらに伸びる余地がある。第 5.4 節の収容人数も同様に**現行設定における値**であって、ハードウェアの上限ではない。

5.3　テール基準で見た飽和点

第 3.5 節で述べたとおり、式 (1) の平均基準は緩い。p99 に制約を課した式 (2) で飽和点を取り直した結果を表 9 に示す。

表 9　平均基準と p99 基準の実用飽和点、および同時利用者数への換算

	平均基準（式 (1)）			p99 基準（式 (2)）		
構成	c_{sat}	ライト	ヘビー	c_{sat}^{p99}	ライト	ヘビー
小モデル ×2	8	10	3	8	10	3
小モデル ×4	32	40	10	16	20	5
小モデル ×8	64	80	20	32	40	10
中モデル ×2	8	10	3	1	2	1
中モデル ×4	16	20	5	4	5	2
中モデル ×8	32	40	10	—	—	—
大モデル ×8	16	20	5	8	10	3

小モデル では収容量がおおむね半分になる。×4 で 32 → 16、×8 で 64 → 32 であり、大モデルも 16 → 8 と半減する。これは平均が飽和点付近でまだ低く保たれている一方、一部のリクエストがすでに待ち行列に入って長く待たされていることを意味する。実運用で「たまに極端に遅い」という苦情を避けたい場合は、**p99 基準の値を採るべき**である。なお 小モデル ×2 は平均基準と p99 基準がともに 8 で一致する。この構成は第 5.6 節のとおり他構成より早く飽和しており、飽和点自体が低いためテールも相対的に穏やかである。

中モデル の落ち込みは半減では済まない。×4 は 16 → 4、×2 は 8 → 1 まで下がり、×8 では**条件を満たす点が存在しない**。×8 は $c = 1$ の時点で TTFT p99 が 6,408 ms に達しており、最小負荷でも 4 秒の制約を超える。中モデル は入力 2048 トークンの prefill 自体に時間がかかるうえ、その所要時間のばらつきが大きいためである。**本稿の負荷条件で「TTFT p99 $\leq$ 4 秒」を要件とするなら、中モデル は事実上候補から外れる**と言ってよい。

一方で 中モデル のこの結果は、平均基準では見えなかった性質を明らかにしている。平均基準では 中モデル ×8 の収容人数は 小モデル ×4 と同等（ライト 40 名）に見えるが、テールまで含めれば両者は比較にならない。**モデル選定にテール要件を持ち込むかどうかで結論が変わる**点に注意が必要である。

なお 小モデル の $c = 2$ では TTFT p99 が 3,944 ms（×4）/ 3,293 ms（×8）と、前後の c（数百 ms 台）に比べて突出している。$c = 2$ の測定は 114 件・118 件と試行数が少なく、単発の外れ値が p99 に乗ったものと考えられる。傾向としては扱わない。

5.4 同時利用者数への換算

実用飽和点 c_{sat} は「サーバが同時に処理しているリクエスト数」であって「同時に利用している人数」ではない。人間の利用には思考・読解・入力のアイドル時間が含まれるため、両者を次の 2 段階で換算する。

1. **アイドル率による割り戻し.** 1 セッションのうち AI が実際に生成している時間の割合を $\rho = 0.8$ とすると、収容できる利用セッション数は c_{sat}/ρ となる。これを**ライト用途**（エージェント並列なし・並行作業なし）の上限とする。
2. **1 人あたり並列数による割り算.** エージェントを最大 2 並列で走らせ、かつ最大 2 件の作業を並行させる**ヘビー用途**では、1 人が同時に $2 \times 2 = 4$ 本の推論を占有する。よってライト用途の上限を 4 で割る。

換算結果を表 10 に示す。

表 10　同時利用者数の見積り（$\rho = 0.8$、ヘビー用途は 1 人 4 並列）

構成	c_{sat}	ライト用途 [人]	ヘビー用途 [人]
小モデル ×2	8	10	3
小モデル ×4	32	40	10
小モデル ×8	64	80	20
中モデル ×2	8	10	3
中モデル ×4	16	20	5
中モデル ×8	32	40	10
大モデル ×8	16	20	5

$\rho = 0.8$ は業務チャットの想定値であり、実測に基づく値ではない。アイドル率は用途によって大きく変わるため、本表は**構成間の相対比較**として読むべきである。また前節のとおり、小モデル・中モデルの値は 同時受け入れ数の上限 8 を前提とした数字である。

5.5 MoE と dense の性能差

同じ GPU 8 枚構成において、活性パラメータ 4B の 小モデル（MoE）は中モデル（dense）の **4.1 倍**の集約スループットを示した（2,907 tok/s 対 705 tok/s）。この差は 2 つの経路で生じている。

1. **1 トークンあたりの計算量.** dense モデルは全パラメータを毎トークン読み出すのに対し、MoE [8] は活性パラメータのみを使う。これは ITL に直接現れており、$c = 1$ の時点で 小モデル が 8.6 ms、中モデル が 23.9 ms と **2.8 倍**の開きがある。
2. **同時実行数の使い切り方.** 第 5.2 節のとおり、小モデル は 同時受け入れ数の上限 8 の枠をレイテンシ制約内で使い切れるのに対し、中モデル は半分（4）しか使えない。同じ設定・同じ台数でも、軽いモデルのほうが枠を活かせる。

一方、大モデル（MoE）は活性 10B と大きく、8 枚構成で 446 tok/s と中モデル（dense, 705 tok/s）を下回った。**MoE であること自体ではなく、活性パラメータ数がスループットを規定する**というのが本試験の示唆である。ただし 大モデル は $c = 1$ のレイテンシが 5.6 秒と、中モデル の 14.2 秒に対して **2.5 倍高速**であり、低負荷での単発応答性では明確に優れている。中モデル（dense）は低負荷でも高負荷でも最も分が悪く、**回答品質を優先する場合にのみ選ぶべき構成**といえる。

5.6 データ並列はレプリカ本数どおりにスケールする

小モデル・中モデルは 1 GPU = 1 レプリカのデータ並列であり、レプリカ間で通信は発生しない。したがって理想的には**レプリカあたりの性能はレプリカ本数によらず一定**になり、集約スループットは本数に正比例するはずである。この予測を検証するため、同時実行数をレプリカ本数で割った**レプリカあたり同時実行数** c/N を横軸に取り直した結果を表 11 に示す。

表 11　レプリカあたりに正規化した性能（小モデル・中モデル、データ並列）

c/レプリカ	小モデル ×2	小モデル ×4	小モデル ×8	中モデル ×2	中モデル ×4	中モデル ×8
レプリカあたりスループット *[req/s]*						
0.5	0.104	0.095	0.095	0.036	0.033	0.032
1	0.172	0.175	0.175	0.065	0.063	0.061
2	0.277	0.304	0.295	0.118	0.108	0.107
4	0.418	0.515	0.487	0.184	0.175	0.174
8	0.427	0.749	0.710	0.229	0.213	0.222
TTFT 平均 *[ms]*						
0.5	367	466	378	1780	2010	2062
1	400	370	366	1989	1923	2199
2	417	396	385	2159	2409	2312
4	539	440	558	2646	3608	3416
8	9357	2011	2586	12153	14184	12784

中モデル は 3 構成が全域で一致し、理想的なデータ並列の挙動を示した。$c/N = 4$ でのレプリカあたりスループットは 0.184 / 0.175 / 0.174 req/s、$c/N = 8$ では 0.229 / 0.213 / 0.222 req/s と、

×2 / ×4 / ×8 のいずれもほぼ同一である。TTFT も揃っており（$c/N = 8$ で 12,153 / 14,184 / 12,784 ms）、レプリカを増やしても 1 本あたりの挙動は変わらない。第 4.5 節で 中モデル が ×1.90 / ×1.99 とほぼ線形にスケールしたのは、この当然の帰結である。

小モデル も ×4 と ×8 は**ほぼ完全に重なる**。$c/N \leq 1$ では小数点以下 3 桁まで一致し、$c/N = 4$ で 0.515 対 0.487、$c/N = 8$ で 0.749 対 0.710 と、負荷が高い領域でも差は 5–8% にとどまる。

ただし 小モデル の ×2 だけが外れている。$c/N = 2$ で 0.277（×4 は 0.304、×8 は 0.295）、$c/N = 4$ で 0.418（同 0.515、0.487）と負荷とともに差が開き、$c/N = 8$ では 0.427 に対し 0.749 / 0.710 と **7 割近い開き**になる。TTFT の差はさらに顕著で、$c/N = 8$ において ×2 が 9,357 ms、×4 が 2,011 ms、×8 が 2,586 ms である。すなわち **×2 構成のレプリカは、同じモデルの他の構成より 1 段早く飽和している**。

データ並列にはレプリカ間の通信がないため、この差を GPU 間トポロジや通信コストで説明することはできない。また 中モデル の ×2 構成は同じ 2 レプリカ構成でありながら他の本数と一致しているので、「レプリカ本数が少ないと不利」という一般的な理由でもない。残る説明は**当該デプロイのレプリカ単体の状態差**である。表 2 のとおり 最大コンテキスト長 8192・GPU メモリ使用率 0.90 に加えて draft モデルが VRAM を占めるため、KV キャッシュの余裕は大きくない。小モデル ×2 の測定時に KV キャッシュが不足し、推論エンジンがシーケンスを退避・再計算していたとすれば、スループットの伸び悩みと TTFT・ITL の悪化が同時に起きることは説明できる。**ただし当時の起動オプションとログは記録が残っておらず、確認できていない。**第 4.5 節の 2 → 4 枚 ×3.58 という超線形も、真の超線形ではなくこの ×2 の低さの裏返しである。

実運用上の含意は明快である。**モデルが GPU 1 枚に収まるなら、データ並列でレプリカを並べる構成は台数効果が素直に得られる**。カード間通信がないためトポロジ（図 1）にも縛られず、NVLink ペアの有無を気にせず任意の枚数へ拡張できる。容量を増やしたい場合、まず検討すべきは同時受け入れ数の上限の引き上げ（第 5.2 節）とレプリカの追加であり、テンソル並列を持ち出す理由はない。

5.7 投機的デコードの寄与

小モデル・中モデルは draft モデルによる投機的デコード [4, 5]（先読み 3 トークン）を有効にして運用されている。小モデル の $c = 1$ における ITL 8.6 ms は利用者 1 人あたり 116 tok/s に相当し、A6000 1 枚で 26B 級のモデルを動かした値としては速い。この単発性能の良さには投機的デコードが寄与していると考えられる。

一方、投機的デコードの利得はバッチが埋まるほど小さくなる。先読みしたトークンが棄却された分の計算は無駄になり、バッチが大きい領域では draft モデルの実行コストのほうが目立つためである。実際、小モデル の ITL は $c/N = 0.25$ の 8.6 ms から $c/N = 8$ の 16.6 ms へと約 2 倍に伸びている。**本稿の ITL はベースモデル単体の復号速度ではなく、投機的デコードを含んだ実運用構成の値である**点に注意されたい。

5.8 大モデルの構成上の制約

大モデル は FP8 でも約 122 GB あり A6000 1 枚（48 GB）に収まらないため、小モデル・中モデルのようなデータ並列は採れず、TP 4 × PP 2 で 8 枚を 1 レプリカとして束ねている。本試験で唯一、**カード間の集団通信が性能に効く構成**である。本機の NVLink はカード 2 枚ずつのペアにし

か張られていないため（図 1）、**TP 4 のグループは必ず PCIe を跨ぐ**。全レイヤの all-reduce が PCIe 帯域に律速され、さらに PP 2 のパイプラインバブルが加わる。

この構成上の不利は測定値にも現れている。$c = 1$ の ITL は 9.76 ms と 小モデル（8.58 ms）に近く、活性 10B のモデルとして妥当な単発性能を示す。ところが負荷をかけると ITL は $c = 32$ で 53.8 ms、$c = 256$ で 259.9 ms へと悪化し、小モデル が $c = 128$ でも 16.7 ms を保つのとは対照的である。

もう一点、1 ステップあたり最大バッチトークン数 4096 の影響が無視できない。本試験の入力長は 2048 トークンであるから、**1 スケジューラステップで prefill できるのは実質 2 リクエスト分**にとどまる。同時受け入れ数の上限 256 と大きな値が設定されていても、prefill 側がこの幅で絞られるため、c が大きい領域では prefill 待ち行列が深くなる。TTFT の悪化（$c = 128$ で 12.8 秒、$c = 256$ で 50.9 秒）と、prefill チャンクが decode ステップに割り込むことによる ITL の悪化は、いずれもこれで説明がつく。なおプレフィックスキャッシュが有効だが、aiperf が生成する合成プロンプトは共通接頭辞をほとんど持たないため、キャッシュヒットによって数値が良く出ている可能性は低い。

改善余地は大きい. TP 2（NVLink ペア内に閉じる）× PP 4 への変更、1 ステップあたりの最大バッチトークン数の拡大、あるいは KV キャッシュを圧迫している 最大コンテキスト長 262144 の削減により、本節で述べたボトルネックはいずれも緩和が見込める。本稿の 大モデル の数値は**この 1 構成における実測値であって、モデルの性能上限ではない**点に注意されたい。

5.9 制約と今後の課題

本試験には以下の制約がある。

1. **閉ループ負荷生成である.** 第 3.5 節のとおり、本試験は常に c 本を飛ばし続ける閉ループ負荷であり、到着率を固定した開ループ負荷とは応答時間分布が定性的に異なる [13]。本稿の飽和点は**同時処理容量**の指標であって、耐えられる到着率（req/s の上限）を保証するものではない。実トラフィックに近い評価にはポアソン到着などの開ループ試験が必要である。
2. **主たる飽和判定が平均基準である.** 式 (1) は TTFT / ITL の平均に制約を課しており、標準的なテール基準 [11, 17] より緩い。第 5.3 節に p99 基準の値を併記したが、そちらでは 小モデル の収容量がおおむね半分になり、中モデル ×8 では条件を満たす点が存在しない。
3. **飽和点が設定値に規定されている.** 第 5.2 節のとおり、小モデル・中モデルの飽和点は同時受け入れ数の上限 8 と一致しており、GPU の計算能力の上限を測れていない。本稿の収容人数は現行設定での値である。
4. **小モデル ×2 の測定値が他構成と整合しない.** 第 5.6 節のとおり、レプリカあたりに正規化しても ×2 のみ性能が低い。KV キャッシュ不足による退避・再計算を疑っているが、当時の推論エンジンの設定とログが残っておらず確認できていない。この構成については再測定が必要である。
5. **大モデル のスケーラビリティが未評価.** 大モデル は 8 枚構成の 1 点のみで、レプリカ本数や並列配置を変えた比較がない。
6. **大モデル は TP 4 × PP 2 の 1 構成のみ.** 第 5.8 節で述べたとおり、この構成には TP グループが PCIe を跨ぐ点と 1 ステップあたり最大バッチトークン数 4096 という 2 つのボトルネックがある。他の並列配置・オプションでの比較データがないため、本稿の数値をモデルの性能上限

と解釈してはならない。

7. **負荷パターンが単一.** ISL 2048 / OSL 512 の 1 条件のみを試験した。要約（長入力・短出力）やコード生成（短入力・長出力）では prefill / decode の比率が変わるため、飽和点も変動する。
8. **アイドル率 $\rho = 0.8$ が想定値.** 第 5.4 節の利用者数換算は実利用ログに基づく値ではない。
9. **クライアント側の上限を測っていないことの確認が未実施.** 負荷クライアントの CPU が飽和していた場合、エンジンではなくクライアントの上限を測ってしまう。ただし 小モデル ×8 で 5.88 req/s まで伸びていることから、少なくともそれ未満の構成でクライアント律速であった可能性は低い。

6 結論

DynaAI-engine に対して、同時実行数を自動でスイープする負荷試験を 7 構成・延べ 52 条件にわたって実施した。得られた知見は次のとおりである。

1. **飽和点の判定には TTFT を用いる.** スループット曲線は飽和後も緩やかに伸び続けるため単独では判断を誤る。TTFT は飽和点で 1 桁跳ね上がり、指標として鋭敏である。
2. **現行の飽和点は 同時受け入れ数の上限 8 が決めている.** 小モデル・中モデルの実用飽和点はレプリカ本数 N に対して $8N$ と一致した。**測っているのはハードウェアの上限ではなく設定の上限**であり、容量を増やしたければまずこの値を見直すべきである。
3. **スループットは活性パラメータ数で決まる.** 活性 4B の 小モデル は 中モデル (dense) の 4.1 倍、活性 10B の 大モデル の 6.5 倍の集約スループットを示した。総パラメータ数でも MoE / dense の別でもない。
4. **データ並列はレプリカ本数どおりにスケールする.** 中モデル は 2 / 4 / 8 枚のいずれもレプリカあたりの性能が一致し、集約スループットは ×1.90 / ×1.99 とほぼ完全に線形であった。小モデル も 4 枚と 8 枚は重なる。1 枚に収まるモデルであれば、データ並列＋ロードバランサで素直に台数効果が得られる。ただし 小モデル ×2 のみ他構成と整合せず、再測定を要する（第 5.6 節）。
5. **収容人数の目安（現行設定）.** 小モデル ×8 枚でライト用途 80 名 / ヘビー用途 20 名、×4 枚で 40 名 / 10 名。中モデル ×8 枚で 40 名 / 10 名、大モデル ×8 枚で 20 名 / 5 名。
6. **モデル選定の指針.** スループット優先なら 小モデル、低負荷での単発応答性なら 大モデル、回答品質を最優先する場合にのみ 中モデル (dense) を選ぶ。ただし 大モデル の高負荷特性は現行の TP 4 × PP 2 構成と 1 ステップあたり最大バッチトークン数 4096 に強く依存しており、再チューニングの余地が大きい（第 5.8 節）。

今後は、(i) 同時受け入れ数の上限を振ってハードウェアの真の上限を測る試験、(ii) 小モデル ×2 の再測定と、大モデル のレプリカ本数を変えた測定、(iii) 大モデル の並列配置・バッチ設定を変えた比較、(iv) 要約・コード生成など負荷パターンを変えた試験、(v) 実利用ログに基づくアイドル率の実測、を課題とする。

参考文献

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, B.-G. Chun, "Orca: A Distributed Serving System for Transformer-Based Generative Models," *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*, 2022.

[3] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. S. Gulavani, A. Tumanov, R. Ramjee, "Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve," *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24)*, 2024.

[4] Y. Leviathan, M. Kalman, Y. Matias, "Fast Inference from Transformers via Speculative Decoding," *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[5] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, J. Jumper, "Accelerating Large Language Model Decoding with Speculative Sampling," arXiv:2302.01318, 2023.

[6] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," arXiv:1909.08053, 2019.

[7] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, Z. Chen, "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] W. Fedus, B. Zoph, N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

[9] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, N. Mellempudi, S. Oberman, M. Shoeybi, M. Siu, H. Wu, "FP8 Formats for Deep Learning," arXiv:2209.05433, 2022.

[10] NVIDIA ai-dynamo, "AIPerf — LLM inference benchmarking tool," https://github.com/ai-dynamo/aiperf（2026 年 7 月時点）.

[11] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, et al., "MLPerf Inference Benchmark," *Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020.（Server シナリオにおける「レイテンシ制約下の最大スループット」の定義を参照した。個別ベンチマークの対象モデルや制約値は版により変わるため、最新の MLPerf Inference 規約を併せて確認のこと。）

[12] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, H. Zhang, "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving," *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24)*, 2024.

[13] B. Schroeder, A. Wierman, M. Harchol-Balter, "Open Versus Closed: A Cautionary Tale," *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '06)*, 2006.

[14] J. D. C. Little, "A Proof for the Queuing Formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[15] A. Georges, D. Buytaert, L. Eeckhout, "Statistically Rigorous Java Performance Evaluation," *Proceedings of the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, 2007.

[16] T. Hoefler, R. Belli, "Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses when Reporting Performance Results," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*, 2015.

[17] J. Dean, L. A. Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.