Explicio

WHITEPAPER · EXPLICIO

Reproduzierbare Entscheidungen auf nicht bitstabiler Hardware

Entscheidungen einer KI-Wissensverwaltung gegen das Rauschen des Inferenzkernels absichern



STAND

Stand 20. September 2026, nach Messungen vom 19. und 20. September 2026

AUTOR

Christian Thiede
TACP Operations GmbH & Co. KG (Explicio), Detmold

Inhalt

1 **Zusammenfassung für Entscheider** 3

2 **Warum das ein Thema ist** 5

3 **Was gemessen wurde** 6

- Umgebung 6
- Werkzeuge 6
- Prompts 7
- Regeln 7

4 **Ergebnisse** 8

- Kein Serving-Schalter beseitigt das Rauschen 8
- Nur Beinahe-Gleichstände kippen 9
- Der Block, der die Experten mischt 10
- Kontrollen 12
- Der Preis eines bitstabilen Betriebs 12

5 **Was Explicio daraus gebaut hat** 14

6 **Lesart für Audits und die KI-Verordnung** 16

7 **Grenzen** 17

8 **Reproduzierbarkeit und Herkunft** 18

- Fassungen und Rohdaten 18
- Über dieses Dokument 18

KAPITEL 1

Zusammenfassung für Entscheider



Ein Sprachmodell, das bei `temperature=0` betrieben wird, gilt gemeinhin als deterministisch: gleiche Eingabe, gleiche Ausgabe. Auf der Hardware, auf der Explicio seine Modelle für EU-Kunden betreibt, stimmt das nicht. Zwanzig identische Aufrufe auf einem leeren Server liefern siebzehn verschiedene Texte. Die Ursache liegt im Rechenkern, der die Experten eines Mixture-of-Experts-Modells im NVFP4-Format mischt; kein Schalter der Serving-Software stellt sie ab, und der batch-invariante Modus des Inferenzservers lehnt genau diesen Kern ab; für die Modellfamilie ist er wegen ihrer Gated-DeltaNet-Schichten außerdem insgesamt nicht verfügbar.

Für eine Wissensverwaltung zählt nicht der Wortlaut einer Antwort, sondern die Entscheidung dahinter: Wird nachgefragt oder nicht? Ist die Eingabe eine Frage oder ein Wissensbeitrag? Welche Kanten bekommt der Wissensgraph? Die Messungen zeigen, wo diese Entscheidungen kippen: nur an Stellen, an denen das Modell zwischen zwei Fortsetzungen fast gleich urteilt. Liegt der Abstand über zwei nats, kippt nichts; in 5.478 gemessenen Positionen ist das kein einziges Mal geschehen.

Explicio hat daraus ein Muster gebaut, das die Entscheidung von der langen Ausgabe trennt: Ein kurzer Aufruf entscheidet, ein langer Aufruf extrahiert. Seitdem sind die Entscheidungen in allen geprüften Betriebsfällen stabil, und die Vollständigkeit der Extraktion wird bei jedem Release gemessen statt vorausgesetzt. Ein bitstabiler Betrieb ist im selben Container auf zwei Wegen möglich: mit fp8-Expertengewichten über den Triton-Kernel zum 1,4- bis 1,8-fachen der Rechenzeit im Einzelstrom, oder mit einem dichten Modell zum 7,4-fachen.

Kennzahl	Wert	Herkunft
Verschiedene Texte in 20 Läufen, Produktionsrezept, leerer Server	17	Leiter, Rohdatei `2026-09-19-leiter.ndjson`
Kipppunkte mit Abstand über 2 nats	0 von 5.478	Logprob-Sonde, alle Zellen
Bitstabile Läufe mit dem dichten Modell (drei Prompts)	60 von 60	Rohdatei `2026-09-19-leiter-2x2.ndjson`
Bitstabile Läufe des MoE-Modells mit fp8-Gewichten (Triton-Kernel)	60 von 60	Rohdatei `2026-09-20-leiter-2x2.ndjson`

Kennzahl	Wert	Herkunft
Stabile Entscheidungen nach dem Umbau (acht Betriebsfälle, je drei Läufe)	8 von 8	`analyze-probe`, Bericht § 7
Erster abweichender Baustein im laufenden Modell	Expertenblock, Schicht 0, Prefill	Rohdatei `opdet-nvfp4-insitu-20260920T095347Z.ndjson`
Rechenzeit der bitstabilen Alternativen im Einzelstrom	fp8 über Triton 1,4-fach (ohne Spekulation) bis 1,8-fach (Produktionsrezept), dicht 7,4-fach	Durchsatz-Rohdateien (`perf.ndjson`)

KAPITEL 2

Warum das ein Thema ist



Die Frage kam aus einem Regressionstest. Ein Fall im Golden-Set von Explicio prüft, dass der Assistent zu einem Widerspruch zwischen zwei Dokumenten keine Fundstelle erfindet. Der Fall war an manchen Tagen grün und an anderen rot, bei unverändertem Code. Die erste Vermutung, ein Fehler in der Prompt-Formulierung, erwies sich als halb richtig: Drei Ursachen lagen tatsächlich in den Prompt-Verträgen und sind behoben. Die vierte lag tiefer. Bei jedem Lauf rechnete das Modell ein wenig anders, und wo zwei Fortsetzungen fast gleich wahrscheinlich waren, gewann mal die eine, mal die andere.

Das ist für eine Wissensverwaltung nicht harmlos. Explicio nimmt Wissen aus Chats, Dokumenten, Bildern und Sprachnachrichten entgegen, zerlegt es in Knoten und Kanten eines Wissensgraphen und beantwortet Fragen mit Belegen aus diesem Graphen. An mehreren Stellen dieser Kette trifft ein Modell eine Entscheidung mit Folgen: Ob eine Eingabe unklar ist und eine Rückfrage auslöst. Ob sie überhaupt Wissen ist oder eine Frage an den Assistenten. Ob zwei Aussagen einander widersprechen. Welche Beziehung zwei Knoten verbindet. Wenn diese Entscheidungen von Lauf zu Lauf schwanken, sieht ein Kunde bei gleicher Eingabe einmal eine Rückfrage und einmal keine, und derselbe Beleg wird einmal als Widerspruch eingetragen und einmal nicht.

Reproduzierbarkeit hat dabei drei Stufen, die man auseinanderhalten muss:

1. **Bitgleich**
   : Zwei Läufe liefern dieselbe Zeichenfolge. Das ist die Stufe, die `temperature=0` verspricht.

2. **Entscheidungsgleich**
   : Zwei Läufe treffen dieselben strukturellen Entscheidungen, auch wenn der Wortlaut abweicht.

3. **Bedeutungsgleich**
   : Zwei Läufe sagen dasselbe, mit anderen Worten.

Für den Betrieb einer Wissensverwaltung ist Stufe 2 die entscheidende. Stufe 1 ist wünschenswert, aber teuer; Stufe 3 ist nicht messbar genug, um ein Tor darauf zu bauen. Dieses Dokument beschreibt, wie weit die Hardware Stufe 1 hergibt, wo Stufe 2 in Gefahr ist und wie Explicio sie sichert.

KAPITEL 3

Was gemessen wurde



Umgebung

Alle Messungen laufen auf der Hardware, die Explicio an Kunden liefert und selbst betreibt: NVIDIA GB10 mit vereinigtem Speicher (Unified Memory, kein getrenntes Grafikspeicher-Konto), vLLM 0.24.0 aus dem NGC-Container 26.07, FlashInfer 0.6.14, CUDA 13.3. Das Produktionsmodell ist `nvidia/Qwen3.6-35B-A3B-NVFP4`, ein Mixture-of-Experts-Modell mit 256 Experten und acht aktiven Experten je Token, dessen Expertengewichte in NVFP4 vorliegen und über den Marlin-Kernel (W4A16, Gruppengröße 16) gerechnet werden; die Projektionen der Aufmerksamkeits- und Gated-DeltaNet-Schichten liegen als statisches FP8 vor. Das Produktionsrezept nutzt Multi-Token-Spekulation (MTP, drei Token), einen fp8-Schlüssel-Wert-Cache, Prefix-Caching und asynchrones Scheduling.

Für die Vergleichszelle ohne Expertenmischung dient `nvidia/Qwen3.6-27B-NVFP4`, das dichte Schwestermodell derselben Familie im selben Format, im selben Container.

Werkzeuge

Vier Werkzeuge, alle ohne Abhängigkeiten außerhalb der Python-Standardbibliothek, alle mit Rohdaten je Lauf:

Die Sonde schickt denselben Prompt zwanzigmal an einen Endpunkt, einmal Lauf für Lauf auf leerem Server (c=1) und einmal mit drei gleichzeitigen Störanfragen (c=4). Sie zählt verschiedene Texte, verschiedene Entscheidungs-Signaturen (die Menge der extrahierten Schlüssel, die Relationstripel, die beiden Flags für Rückfrage und Frage) und die Position der ersten Abweichung.

Die Leiter fährt Serving-Konfigurationen mit je einer Änderung gegenüber der vorigen Stufe durch dieselbe Sonde: ohne Prefix-Cache, ohne Spekulation, mit bf16-Cache, mit allen Abzügen zusammen, mit dem batch-invarianten Modus, mit dem dichten Modell. Jede Stufe läuft auf einer Box, die zuvor aus dem Serving-Pool genommen und deren Produktionsprofil angehalten wurde; die Box teilt sich mit nichts.

Die Logprob-Sonde lässt sich zu jedem erzeugten Token die zwanzig wahrscheinlichsten Kandidaten mit ihren Log-Wahrscheinlichkeiten ausgeben. Sie richtet die Läufe auf Token-Ebene aneinander aus, bestimmt an jeder Abweichung die Marge (Logprob des gewählten Tokens minus Logprob des zweitbesten) und misst im gemeinsamen Präfix zweier Läufe das Rauschen: die Differenz der Logprobs desselben Tokens an derselben Position bei identischem Kontext.

Der Operator-Harness ruft die einzelnen Rechenbausteine des Modells im Container isoliert auf (Normalisierung, Router, Gated-DeltaNet-Kern, Aufmerksamkeit, FP8-Linear) und vergleicht fünf Wiederholungen bitweise, einmal mit gleichem Batch und einmal mit angehängten Störzeilen.

Prompts

Gemessen wurde mit zwei Prompt-Sätzen. Der Betriebs-Prompt ist der Extraktionsvertrag, den Explicio im Produkt verwendet, mit einer Sachnotiz von 1.881 Token über zwei Dokumente, die einander im Betriebsdruck einer Pumpe widersprechen. Der öffentliche Satz besteht aus zwanzig synthetischen Sachnotizen (zehn deutsch, zehn englisch, 40 bis 80 Wörter, ohne Produkt-, Kunden- oder Personendaten) mit einem neutralen Extraktionsvertrag; er wird mit den Werkzeugen und Rohdaten veröffentlicht, damit die Ergebnisse auf anderer Hardware nachgestellt werden können.

Regeln

Jede Zahl in diesem Dokument hat eine Rohdatei. Jedes Ergebnis, das eine Empfehlung trägt, ist wiederholt gemessen. Ein Kandidat, der nicht startet, ist ein Ergebnis und steht in der Tabelle. Und weil eine Messung, die dieselbe Konfiguration einmal grün und einmal rot zeigt, ohne Wiederholung nichts beweist, wird sie zwanzigmal je Stufe gefahren, bevor jemand an einer Formulierung dreht.

KAPITEL 4

Ergebnisse



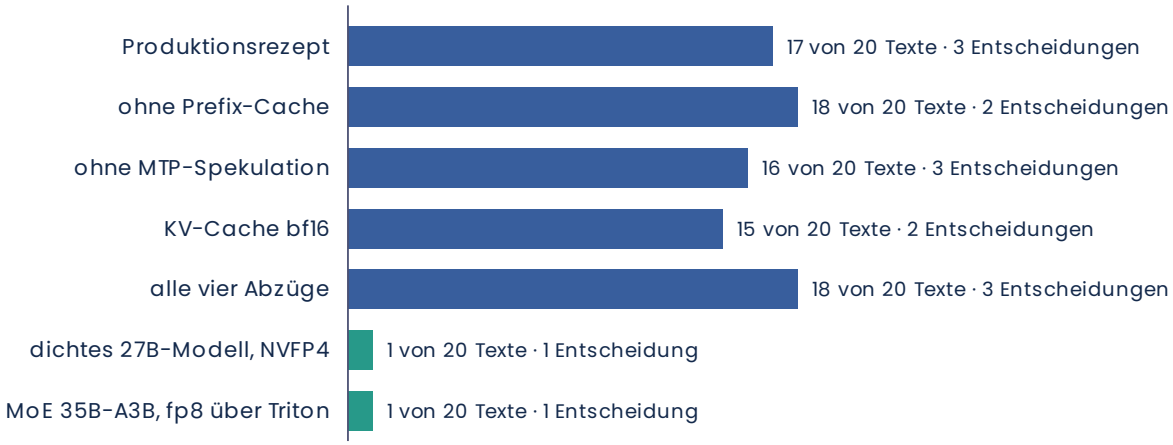
Kein Serving-Schalter beseitigt das Rauschen

Die Leiter beantwortet die erste Frage: Liegt es an einer Einstellung des Servers? Sie tut es nicht. Jede Stufe entfernt eine Funktion, die Rechenwege verändern könnte, und keine Stufe ändert das Bild.

Konfiguration	Verschiedene Texte (20 Läufe, leerer Server)	Verschiedene Entscheidungen	Erste Abweichung (Median, Zeichen)
Produktionsrezept	17	3	171
ohne Prefix-Cache	18	2	175
ohne MTP-Spekulation	16	3	171
Schlüssel-Wert-Cache bf16	15	2	171
alle vier Abzüge zusammen	18	3	171
batch-invarianter Modus (`VLLM_BATCH_INVARIANT=1`)	startet nicht	entfällt	entfällt
dichtes 27B-Modell, NVFP4	1	1	keine
MoE 35B-A3B, fp8-Gewichte über den Triton-Kernel	1	1	keine
MoE 35B-A3B fp8, volles Produktionsrezept (Spekulation, fp8-Cache)	1	1	keine

Beleg: `docs/messungen/determinism/2026-09-19-leiter.ndjson`, `2026-09-19-leiter-2x2.ndjson` und `2026-09-20-leiter-2x2.ndjson`, Tabelle über `tools/bakeoff/determinism-report.py`.

ABB. 1 **Verschiedene Texte je Konfiguration**



1 Text = bitstabil. Rohdaten: 2026-09-19-leiter.ndjson, 2026-09-19/20-leiter-2x2.ndjson

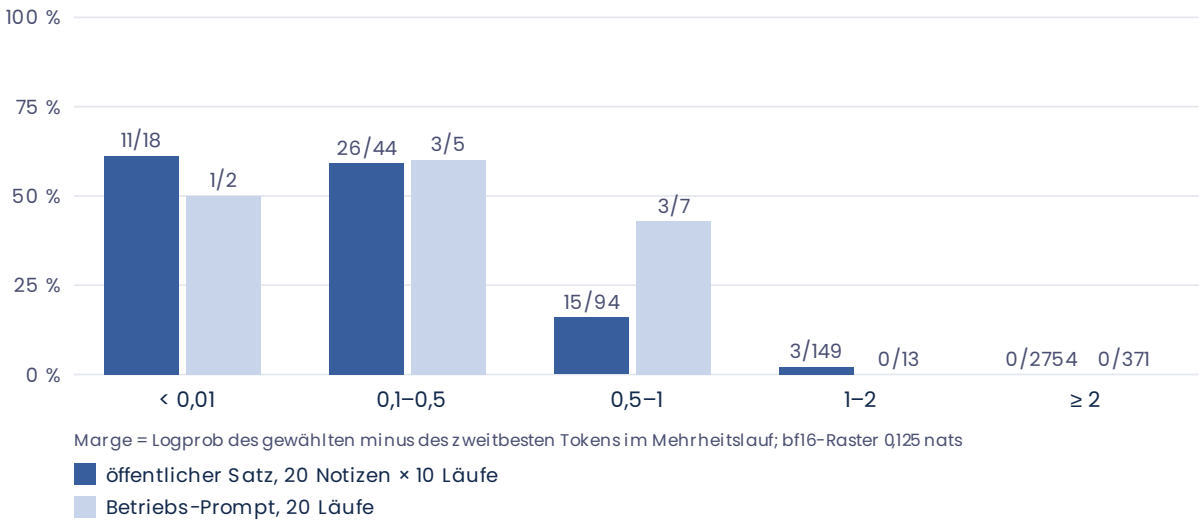
Der batch-invariante Modus startet nicht, und der Fehlertext sagt, warum: `NvFp4 MoE backend 'MARLIN' does not support the deployment configuration since kernel does not support batch invariance`. Im Container erklärt nur ein einziger Expertenkernel Invarianz, der Triton-Kernel für fp8-Gewichte; alle NVFP4-Kernel lehnen ab. Mit fp8-Gewichten wird dieser Kernel gewählt, und der Modus scheitert eine Ebene früher: `VLLM batch_invariant mode is not supported for GDN_ATTN`, am Aufmerksamkeits-Backend der Gated-DeltaNet-Schichten. Für die hybride Modellfamilie ist der Modus damit unabhängig vom Gewichtsformat nicht verfügbar.

Die Entscheidungen bleiben über alle 200 Leiter-Läufe hinweg in dem, was für das Produkt zählt, stabil: Die Kante `contradicts` zwischen den beiden Dokumenten wird in 200 von 200 Läufen gesetzt, drei Knoten werden in 200 von 200 Läufen erkannt, eine Rückfrage in 0 von 200 Läufen gestellt. Was schwankt, ist ein Kantentyp (36 von 200 Läufen `documents` statt `applies_to`) und die Schreibweise eines Schlüssels (5 von 200).

Nur Beinahe-Gleichstände kippen

Die Logprob-Sonde macht sichtbar, was die Leiter nur zählt. An jeder Position, an der zwei Läufe voneinander abweichen, hat das Modell im Mehrheitslauf zwischen dem gewählten und dem zweitbesten Token eine Marge. Diese Marge ist an Kipppunkten klein und an stabilen Positionen groß: Im Betriebs-Prompt liegt sie an stabilen Positionen im Median bei 12,2 nats, an den sieben Kipppunkten im Median bei 0,375 nats und nie über 0,875 nats.

ABB. 2 **Kipprate je Margen-Klasse**



Auf dem öffentlichen Satz mit 20 Notizen und je 10 Läufen verteilen sich die Kipppunkte so: Unter 0,01 nats kippen 11 von 18 Beinahe-Gleichständen, zwischen 0,1 und 0,5 nats 26 von 44, zwischen 0,5 und 1 nat 15 von 94, zwischen 1 und 2 nats 3 von 149, und über 2 nats 0 von 2.754. Der Betriebs-Prompt zeigt dasselbe Bild mit 0 von 371 über 2 nats. Die Klassen zwischen 0,01 und 0,1 sind leer, weil die Schnittstelle Log-Wahrscheinlichkeiten in bf16-Auflösung liefert: Alle Werte sind Vielfache von 0,125 nats, eine Marge unter 0,125 ist auf dieser Skala ein exakter Gleichstand.

Beleg: `2026-09-19-logprobs-public.ndjson` (20 Zellen) und `2026-09-19-logprobs.ndjson`, Tabelle über `determinism-report.py --margins`.

Wie groß ist das Rauschen? Im gemeinsamen Präfix zweier Läufe, also bei identischem Kontext, unterscheiden sich die Log-Wahrscheinlichkeiten des zweitbesten Tokens im Median um 0,375 nats, im 90. Perzentil um 1,125 nats, im Maximum um 11,5 nats. Das gewählte Token selbst zeigt fast kein Rauschen (Median $8 \cdot 10^{-6}$), was die Log-Softmax erklärt: Ihr Normierer wird vom Spitzen-Token dominiert, dessen eigenes Rauschen sich herauskürzt. Die Ausreißer von mehreren nats sind diskrete Sprünge, wie sie entstehen, wenn in einer Schicht ein anderer Experte gewählt wird. Rundungsrauschen bliebe innerhalb von ein, zwei bf16-Inkrementen.

Die Kippgrenze und das Rauschen passen zusammen: Gekippt wird, wo die Marge unter das Rauschen fällt. Ein Beinahe-Gleichstand kippt nicht sicher, aber wahrscheinlich: Auf dem öffentlichen Satz hielten 157 Beinahe-Gleichstände, 55 kippten.

Der Block, der die Experten mischt

Die zweite Frage lautet: Welcher Baustein rauscht? Zwei Messungen antworten von zwei Seiten.

Das dichte 27B-Modell derselben Familie, im selben NVFP4-Format, im selben Container, mit demselben Marlin-Kernel für seine Gewichte, liefert auf drei Prompts 60 von 60 Läufen bitidentisch. Die Logprob-Sonde findet an 1.054 Positionen keinen einzigen Unterschied von einem bf16-Inkrement, und drei exakte Gleichstände (Marge 0 auf dem bf16-Raster) fallen 20 von 20 Mal auf dieselbe Seite. Das geht nur, wenn die Logits bitgleich sind. In der dichten Zelle rechnen dieselben Bausteine wie im Produktionsmodell (Dequantisierung und Matrixmultiplikation des NVFP4-Kernels, FlashInfer-Aufmerksamkeit, die Gated-DeltaNet-Schichten, Normalisierung, Sampler), mit einer Ausnahme: dem Block, der je Token acht von 256 Experten wählt, die Eingabe auf sie verteilt und ihre Ergebnisse gewichtet zusammenführt.

Dasselbe MoE-Modell mit fp8-Expertengewichten, gerechnet über den Triton-MoE-Kernel ohne jeden Invarianz-Schalter, ist auf denselben drei Prompts 60 von 60 Läufen bitidentisch, mit Logit-Rauschen null an 875 Positionen. Das Rauschen ist also kein Merkmal der Expertenmischung als solcher, sondern des NVFP4-Marlin-Pfads. Das gilt auch im vollen Produktionsrezept mit Spekulation, fp8-Cache, Prefix-Cache und asynchronem Scheduling: 60 von 60 bitidentisch, ebenso unter Störlast. Das dichte Modell in fp8 ist auf drei Prompts bitstabil, auch unter Störlast. Eine Wiederholung der fp8-MoE-Zelle nach vollständigem Neustart des Servers lieferte denselben Text wie der erste Lauf, 20 von 20 in beiden Läufen; die Stabilität hält also über Prozessgrenzen hinweg. Der In-situ-Harness auf dem fp8-Modell zeigt dasselbe auf Aktivierungsebene: Zwischen zwei Läufen weicht keines der 251 Module ab, und in der Batch-Variante bleiben alle 40 Schichten bitgleich; erst die Logit-Projektion am Ende reagiert auf die andere Batch-Höhe. Batch-Abhängigkeit trat auf der fp8-Schiene einmal auf: Die Zelle ohne Spekulation lieferte auf dem Betriebs-Prompt unter Störlast zwei Texte in zusammenhängenden Blöcken, in den fünf anderen fp8-Zellen unter Störlast blieb es bei einem Text.

Der Operator-Harness bestätigt das für die Bausteine, die sich isoliert aufrufen lassen. Normalisierung, Router mit Top-8-Auswahl, Gated-DeltaNet-Kern, FlashInfer-Prefill und die FP8-Matrixmultiplikation sind bei 1.881 Token in fünf Wiederholungen und unter angehängten Störzeilen bitgleich, mit und ohne Invarianz-Schalter (maximale Abweichung 0 in jeder Zelle). Der Expertenkernel selbst lässt sich nicht isoliert aufrufen, weil er die umgepackten Gewichte einer echten Modellschicht braucht.

Den Nachweis im laufenden Modell liefert der zweite Modus des Harness: Hooks an allen 251 Modulen des Produktionsmodells zeichnen bei zwei Läufen desselben Prompts jede Ausgabe auf und vergleichen sie in Ausführungsreihenfolge. Der erste Tensor, der abweicht, ist die Ausgabe des Expertenblocks der ersten Schicht im Prefill. Davor sind Eingangs- und Nach-Aufmerksamkeits-Normalisierung, die Router-Logits und der geteilte Experte derselben Schicht bitgleich; danach weicht jede folgende Schicht ab. Die Abweichung ist winzig: 54 von 3.852.288 Elementen (1.881 Token × 2.048 Dimensionen) unterscheiden sich, um höchstens $6{,}1 \cdot 10^{-5}$ absolut, was bei kleinen Werten bis zu 25 Prozent relativ ist. Ein Batch mit einer angehängten Störanfrage zeigt dieselbe erste Abweichung im selben Modul.

Beleg: `2026-09-19-leiter-2x2-logprobs.ndjson` (die dichten Zellen), die Dateien `opdet-ops` und `opdet-ops-invariant` sowie `opdet-nvfp4-insitu-20260920T095347Z.ndjson` im Ordner `opdet`.

Eine Einschränkung gehört dazu. Unter drei gleichzeitigen Störanfragen liefert das dichte Modell auf dem Betriebs-Prompt zwei Texte, aber nicht zufällig verteilt: Die Läufe 1 bis 9 sagen an Zeichen 782 „gibt … an“, die Läufe 10 bis 20 „weist … aus“. Das ist die Abhängigkeit von der Batch-Zusammensetzung, die der batch-invariante Modus adressiert. Das dichte Modell ist also bei fester Batch-Form bitstabil, aber nicht batch-invariant.

Kontrollen

Drei Einwände lagen nahe, und alle drei sind gemessen.

Erster Einwand, die Spekulation: Das Produktionsrezept rät mit einem Zusatzmodell drei Token voraus und lässt sie vom Hauptmodell prüfen; dieser Prüfschritt könnte die Logprobs prägen. Dieselbe Sonde gegen dasselbe Modell ohne Spekulation zeigt dieselbe Kippgrenze: 9 von 9 Kipppunkten unter dem Rausch-p99, über 2 nats 0 von 798 Positionen. Mit Spekulation: 13 von 13, 0 von 800.

Zweiter Einwand, die zwei Boxen: Die ersten Messungen gegen den Produktionsendpunkt liefen über zwei Boxen im Pool. Die Kontrolle auf einer einzelnen Box im Messfenster ändert nichts am Bild.

Dritter Einwand, der Vertrag: Der Betriebs-Prompt trägt die Härtungen aus dem nächsten Abschnitt. Der öffentliche Satz mit einem neutralen Extraktionsvertrag zeigt, was ohne sie passiert: Bei 400 Läufen über 20 Notizen weichen 26 Läufe in einem der beiden Flags (Rückfrage, Frage) von der Mehrheit ab, und 92 Läufe in der Struktur (Schlüsselmenge oder Relationen). Das sind 6,5 Prozent und 23 Prozent. Kernel und Server sind in beiden Messungen dieselben; nur der Vertrag unterscheidet sich.

Beleg: `2026-09-19-leiter-2x2-logprobs.ndjson` (die Zellen `det-prod-pub` und `det-plain-pub`), `2026-09-19-public-ref.ndjson`.

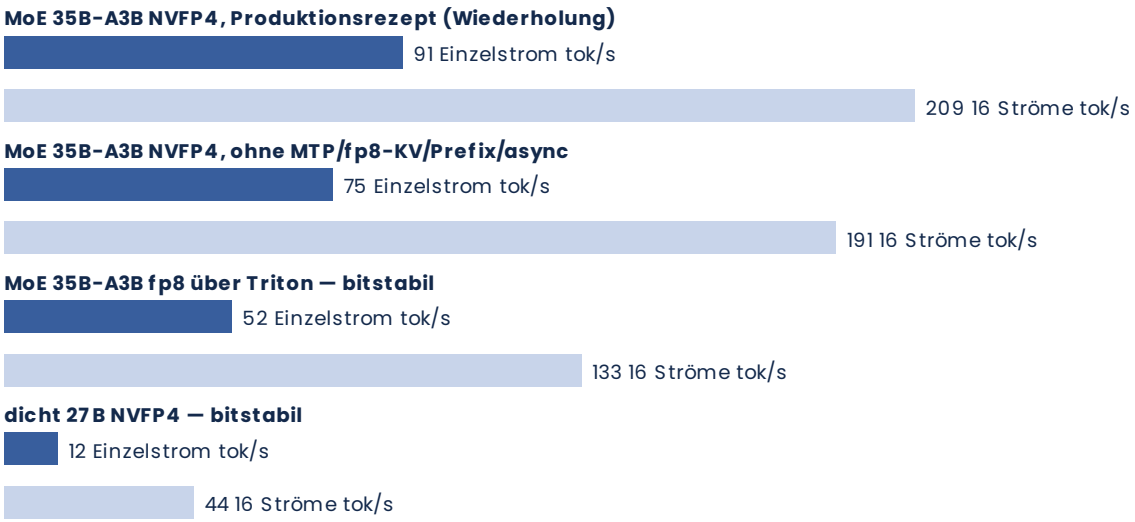
Der Preis eines bitstabilen Betriebs

Bitstabilität ist im heutigen Container mit dem dichten Modell zu haben, und der Preis lässt sich auf derselben Box beziffern.

Konfiguration	TTFT (s)	Einzelstrom	16 Ströme	Prefill
MoE 35B-A3B NVFP4, Produktionsrezept	0,130	91,5	209	5.420
MoE 35B-A3B NVFP4, ohne Spekulation, Cache und Prefix	0,095	75,4	191	5.556
MoE 35B-A3B fp8 über Triton, ohne Spekulation und Cache, bitstabil	0,146	52,2	133	3.414
MoE 35B-A3B fp8 über Triton, Produktionsrezept, bitstabil	0,206	51,3	143	3.231
dicht 27B NVFP4, bitstabil	0,308	12,3	43,6	920
dicht 27B fp8, bitstabil	0,401	7,8	29,4	366

Durchsatz in Token je Sekunde; TTFT = Zeit bis zum ersten Token. Beleg: `2026-09-19-det-prod-pub-perf.ndjson`, `2026-09-19-det-plain-pub-perf.ndjson`, `2026-09-19-det-dense-nvfp4-perf.ndjson`, `2026-09-20-det-moe-fp8-plain-perf.ndjson`, `2026-09-20-det-moe-fp8-perf.ndjson`, `2026-09-20-det-dense-fp8-perf.ndjson`; Messwerkzeug `tools/bakeoff/perf.py`, 256-Token-Prompts.

ABB. 3 **Durchsatz je Konfiguration**



Rohdaten: det-{prod-pub,plain-pub,dense-nvfp4,moe-fp8-plain}-perf.ndjson (19./20.09.2026)

Die fp8-Variante desselben MoE-Modells über den Triton-Kernel kostet bei gleicher Funktion, also Produktionsrezept gegen Produktionsrezept, 44 Prozent des Einzelstrom-Durchsatzes, 31 Prozent bei 16 Strömen und 40 Prozent beim Prefill, dazu rund 12 GB mehr Speicher für die Gewichte. Ohne Spekulation und Cache auf beiden Seiten sind es 31, 30 und 39 Prozent; die Spekulation, die dem NVFP4-Rezept im Einzelstrom 21 Prozent bringt, bringt dem fp8-Rezept dort nichts. Das dichte Modell braucht das 7,4-fache und bei 16 Strömen das 4,8-fache an Zeit, weil es 27 Milliarden Parameter je Token rechnet statt drei, und die dichte fp8-Variante ist noch langsamer. Die fp8-Zellen waren im ersten Messfenster an einem Warmup-Fehler des Containers gescheitert (DeepGEMM, `Unknown recipe`, obwohl vLLM DeepGEMM für das Modell selbst abgeschaltet hatte) und liefen im zweiten Fenster mit abgeschaltetem DeepGEMM.

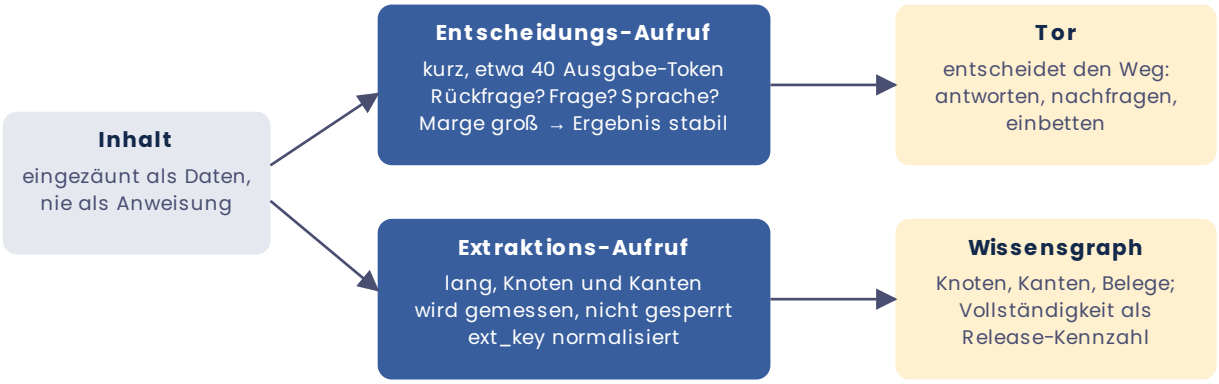
Eine Zahl in der Tabelle ist selbst ein Ergebnis der Methode: Die erste Messung des Produktionsrezepts hatte 3.288 Token/s Prefill ergeben, die Wiederholung 5.420. Die erste Messung lief unmittelbar nach einem Neustart der Box und war nicht belastbar; in der Tabelle steht die Wiederholung.

KAPITEL 5

Was Explicio daraus gebaut hat

Die Konsequenz ist ein Muster auf der Vertragsebene, das die Entscheidung dorthin verlegt, wo das Rauschen sie nicht erreicht; Modell und Kernel bleiben unverändert.

ABB. 4 **Entscheidung kurz, Extraktion lang**



Das Rauschen des Kernels trifft die lange Ausgabe; die Entscheidung liegt in einem Aufruf, dessen Marge es nicht erreicht.

Entscheidung kurz, Extraktion lang. Der Analyse-Schritt stellt dem Modell zwei getrennte Aufgaben. Ein kurzer Aufruf mit etwa vierzig Ausgabe-Token beantwortet drei Fragen: Ist eine Rückfrage nötig, ist die Eingabe eine Frage an den Assistenten, in welcher Sprache ist sie geschrieben. Ein langer Aufruf extrahiert Knoten, Kanten und Belege. Die Entscheidung des kurzen Aufrufs steuert den Weg; die Extraktion füllt den Graphen. Eine kurze Ausgabe hat wenige Positionen, an denen ein Beinahe-Gleichstand entstehen kann, und eine Ja-Nein-Entscheidung mit klarer Regel hat an ihrer Position eine große Marge. Die Messung nach dem Umbau: In acht Betriebsfällen mit je drei Läufen ist die Entscheidung achtmal stabil.

Ein Vertrag verlangt nur, was es gibt. Drei der vier Ursachen des Golden-Falls lagen in den Verträgen selbst: Ein nacktes Speicherdatum im Widerspruchs-Block wurde zur Seitenzahl, ein Etikett „keine Fundstelle“ zur Klammer hinter dem Wert, und der Marker „S.“ aus der Zitierregel landete in „(S. 180 bar)“, wo nichts zu zitieren war. Seitdem reist jeder Wert beschriftet in den Prompt, und die Zitierregel wird nur eingefügt, wenn eine Fundstelle existiert. Ein Vertrag, der Seitenzahlen verlangt, bekommt sie auch dort, wo es keine gibt.

Schlüssel werden normalisiert. Die Schreibweise eines `ext_key` (5 von 200 Läufen) ist kein Modellproblem, sondern eine fehlende Regel: Der Ingestion-Pfad normalisiert Schlüssel jetzt kanonisch, bevor sie in den Graphen gehen.

Vollständigkeit wird gemessen, nicht gesperrt. Die Extraktion flattert weiterhin: In vier von acht Betriebsfällen weicht die Zahl der Kanten oder Knoten zwischen drei Läufen ab. Ein Tor, das daran rot würde, wäre auf diesem Kernel immer rot. Deshalb zählt die Sonde die Abweichung und schreibt sie in jedes Modell-Release, damit ein Wechsel der Gewichte oder des Containers sie sichtbar bewegt. Das Release-Tor prüft zusätzlich, dass die Entscheidungen über drei Läufe stabil bleiben.

Zwei Regeln sind daraus in die Baurichtlinien von Explicio gewandert: Ein Entscheidungs-Aufruf darf nicht auf einer Vertragsgrenze sitzen, weil dort das Rauschen entscheidet. Und wo ein Prompt dreimal dieselbe Sache falsch macht, ist das Format schuld, nicht die Formulierung.

KAPITEL 6

Lesart für Audits und die KI-Verordnung



Explicio ist nach eigener Einstufung kein Hochrisiko-KI-System im Sinne der Verordnung (EU) 2024/1689; die Einstufung, die Herleitung und jede Neubewertung stehen im KI-Systemregister des Unternehmens. Die Pflichten zu Aufzeichnung und Nachvollziehbarkeit aus Artikel 12 gelten deshalb nicht unmittelbar. Die Messungen in diesem Dokument beantworten trotzdem eine Frage, die jeder Auditor stellen kann und die ohne Messung nur mit einer Behauptung zu beantworten wäre: Was heißt „reproduzierbar“ für dieses System, technisch?

Die Antwort in drei Sätzen: Auf der Hardware, die Explicio betreibt, ist die Zeichenfolge einer Antwort bei gleicher Eingabe nicht garantiert. Die strukturellen Entscheidungen (Rückfrage, Frage, Sprache und die Kanten, die das Produkt für seine Belege braucht) sind es nach dem Umbau in allen geprüften Fällen. Die Vollständigkeit einer Extraktion ist es nicht, und sie wird bei jedem Release beziffert.

Für die Nachvollziehbarkeit einzelner Antworten trägt jede Ausgabe des Assistenten eine Kennzeichnung mit Herkunft, Modell, Prompt-Fassung, Route und den verwendeten Quellen; damit lässt sich zu jeder Antwort sagen, welches Modell in welcher Fassung sie erzeugt hat und aus welchen Belegen. Der Modellname steht im Transparenzblatt, jeder Kanal trägt den Hinweis nach Artikel 50, und ein Modellwechsel ist ein Eintrag im Modellregister im selben Commit wie die Konfiguration.

Die Veröffentlichung dieser Messungen ändert die Einstufung nicht. Sie beschreibt das System; sie trainiert kein Modell, verändert keine Gewichte und bewertet keine Person.

KAPITEL 7

Grenzen



- Die fp8-Zellen sind auf drei Prompts mit je 20 Läufen gemessen, die Wiederholung auf dem Betriebs-Prompt. Der batch-invariante Modus des Servers bleibt für diese Modellfamilie unmessbar, weil er die Gated-DeltaNet-Schichten nicht unterstützt; die Aussage, dass nur der Triton-Kernel Invarianz erklärt, ist eine Aussage über den Quelltext.
- Der In-situ-Nachweis benennt das Modul, nicht die Zeile im Kernel. Ob die Verteilung der Token auf die Experten, die Matrixmultiplikation je Experte oder die Zusammenführung der Expertenergebnisse den Unterschied erzeugt, ist offen; das ist die Frage, mit der ein eigener, invarianter Kernel beginnen müsste.
- Eine Hardware, ein Container, eine Modellfamilie. Alle Zahlen gelten für GB10, vLLM 0.24.0 aus NGC 26.07 und Qwen3.6 in NVFP4. Ob andere Beschleuniger, andere Kernel oder andere Quantisierungen dasselbe Bild zeigen, sagt dieses Dokument nicht.
- Die Durchsatzzahlen sind Einzelmessungen mit einer Wiederholung. Die Wiederholung hat einen Ausreißer entlarvt; weitere Wiederholungen würden die Streuung beziffern, die hier nur als Warnung steht.
- Der öffentliche Prompt-Satz ist synthetisch. Er ist so gebaut, dass er die Klasse von Ausgaben des Betriebs erzeugt, aber er ist nicht der Betrieb.
- Log-Wahrscheinlichkeiten in bf16-Auflösung. Margen unter 0,125 nats sind auf dieser Skala nicht auflösbar; die Klasse „unter 0,01“ heißt in Wahrheit „Gleichstand“.
- **Die Aussage „über 2 nats kippt nichts“ ist eine Beobachtung über 5.478 Positionen**, keine Garantie. Ein Ausreißer im Rauschen von 11,5 nats ist gemessen; er traf eine Position mit großer Marge, nicht den Kipppunkt.

KAPITEL 8

Reproduzierbarkeit und Herkunft



Fassungen und Rohdaten

Bestandteil	Fassung
Inferenzserver	vLLM 0.24.0+092c4842, Container `nvcr.io/nvidia/vllm:26.07-py3`
Beschleuniger	NVIDIA GB10, CUDA 13.3, FlashInfer 0.6.14, torch 2.13.0a0+nv26.07
Produktionsmodell	`nvidia/Qwen3.6-35B-A3B-NVFP4`
dichtes Vergleichsmodell	`nvidia/Qwen3.6-27B-NVFP4`, Revision `0893e160…`
fp8-Checkpoints	`Qwen/Qwen3.6-35B-A3B-FP8` Revision `95a723d0…`, `Qwen/Qwen3.6-27B-FP8` Revision `e89b16eb…`
Werkzeuge	`tools/bakeoff/determinism-probe.py`, `determinism-ladder.sh`, `determinism-logprobs.py`, `determinism-report.py`, `opdet/harness.py`
Rohdaten	alle `.ndjson`-Dateien unter `docs/messungen/determinism/` und im Ordner `opdet/`; Positionen als `.positions.ndjson.gz`
Öffentlicher Prompt-Satz	`docs/messungen/determinism/public-prompts/`, CC BY 4.0
Vollständiger Messbericht	`docs/messungen/determinism/2026-09-19-leiter.md` (§ 1 bis § 10)

Werkzeuge, öffentlicher Prompt-Satz und die Rohdaten der öffentlichen Läufe erscheinen unter Apache-2.0 in einem eigenen Repository, mit einem Aufruf, der die Messung gegen einen beliebigen OpenAI-kompatiblen Endpunkt wiederholt. Der Betriebs-Prompt bleibt intern; dieses Dokument beschreibt ihn strukturell.

Über dieses Dokument

Autor: Christian Thiede, TACP Operations GmbH & Co. KG (Explicio), Detmold, ORCID 0009-0002-4517-7208. Bei Messung, Auswertung und Text hat ein KI-Assistent (Claude, Anthropic) mitgearbeitet; die Verantwortung für Inhalt und Zahlen liegt beim Autor. Aussagen über Software Dritter (vLLM, FlashInfer, DeepGEMM, NVIDIA-Container, Qwen-Modelle) beziehen sich auf die genannten Fassungen und das Datum der Messung. Das Dokument darf unverändert weitergegeben werden.



Explicio®

Das digitale Gedächtnis Ihres Unternehmens

TACP Operations GmbH & Co. KG (Explicio) · Detmold