Zakaj grafični pomnilnik določa, kaj lahko poganjamo

Kako uteži modela, natančnost, kontekst in sočasne zahteve porabljajo pomnilnik?



POMNILNIK IN ZMOGLJIVOST

Izračunan primer potrebne zmogljivosti, ki loči predpostavke od meritev na strojni opremi.

Viri preverjeni 25. septembra 2026.

Jezikovni model se lahko uspešno naloži, odgovori na kratko vprašanje, nato pa mu zmanjka grafičnega pomnilnika, ko ga začnejo uporabljati še sodelavci. Uspešno nalaganje potrdi le, da je uspel en korak. Ne potrdi, da lahko računalnik hrani več dolgih pogovorov in hkrati pripravlja odgovore. Uporabna ocena potrebnega pomnilnika zato opisuje obremenitev, ne le imena modela.

Članek 09 je primerjal lokalno opremo, najete strežnike in gostovane storitve. Ta članek podrobneje razloži izračun pomnilnika, na katerem temelji takšna izbira. Izračunani primer je izmišljen in namenoma preprost. Številke so računsko preverjene, vendar nismo prenesli modela ali preizkusili GPU-ja. Rezultat ni priporočilo za nakup.

Začnite pri utežeh, vendar se tam ne ustavite

Uteži so naučene številske vrednosti, ki jih model uporablja pri sklepanju, torej pri ustvarjanju izhodov z že naučenim modelom. Če so vsi parametri shranjeni v isti obliki, njihovo idealno velikost dobimo tako, da število parametrov pomnožimo s številom bitov na parameter in delimo z osem. To je izračun velikosti tenzorjev. Ne vključuje vsebnika, pomožnih podatkov kvantizacije ali vsega, kar izvajalno okolje potrebuje med obdelavo zahtev.

Predpostavimo natanko osem milijard shranjenih parametrov. Spodnja tabela upošteva samo njihovo idealno velikost, pri čemer ima vsak parameter navedeno bitno širino. Ne prikazuje izmerjenih velikosti datotek modelov ali zahtev za napravo.

Predpostavljena bitna širina parametra	Velikost v decimalnih GB	Velikost v binarnih GiB
32 bitov	32	29,80
16 bitov	16	14,90
8 bitov	8	7,45
4 biti	4	3,73

Enote so pomembne. En GB pomeni 1.000.000.000 bajtov, en GiB pa 1.073.741.824 bajtov. NIST objavlja to razlikovanje med decimalnimi in binarnimi predponami.[5] S pretvarjanjem enot podatkov ne stisnemo. Istih šestnajst milijard bajtov lahko izrazimo kot 16 GB ali približno 14,90 GiB. V izračunu uporabljajte eno enoto ter preverite, kaj dejansko pomenijo navedbe v specifikaciji naprave in orodju za spremljanje. Njune oznake niso nujno zamenljive.

Natančnost shranjevanja in natančnost računanja sta ločeni nastavitvi. Sistem lahko uteži hrani stisnjene, računa pa z večjo natančnostjo. Dokumentacija Hugging Face za bitsandbytes ločeno opisuje kvantizirane linearne plasti, podatkovni tip drugih modulov in podatkovni tip računanja pri štiribitni kvantizaciji.[7]

Oznaka »štiribitni model« zato ne zadošča za opredelitev preizkusa pomnilniških zahtev. Zabeležite točno datoteko modela, metodo kvantizacije, izvajalno okolje in pomembne podatkovne tipe. Upoštevajte dodatne podatke, ki jih zahteva izbrana oblika zapisa, in komponente z večjo natančnostjo. Zadnja vrstica tabele ni celotna velikost modela v pomnilniku.

Ta izračun ne obravnava učenja. Učenje spreminja parametre in lahko zahteva gradiente, stanja optimizatorja ter shranjene aktivacije za povratni prehod. Vodnik Hugging Face o porabi pomnilnika opisuje te potrebe pri učenju, vključno z ločenimi tenzorji optimizatorja in začasnimi konicami med operacijami.[1] Ocene za sklepanje ne smemo predstavljati kot zahtevo za učenje. Prav tako pravila o bajtih na parameter pri učenju ne smemo prenesti v izračun sklepanja brez pojasnila, kaj vključuje.

Skupno število parametrov ni število aktivnih parametrov

Model z mešanico ekspertov, običajno označen kot MoE, usmerja izračun skozi izbrani del svojih ekspertnih komponent. Modelna kartica Qwen3-30B-A3B navaja skupno 30,5 milijarde parametrov in 3,3 milijarde aktiviranih parametrov, pri čemer je aktiviranih osem od 128 ekspertov.[3] To sta različni količini, ne dva nasprotujoča si opisa velikosti datoteke.

Za ponazoritev predpostavimo dva bajta za vsak shranjeni parameter. Pri 30,5 milijarde parametrov to pomeni 61 GB. Če pomnožimo samo objavljeno število aktivnih parametrov, dobimo 6,6 GB. Drugi rezultat ni ocena velikosti vseh uteži: izpusti

parametre komponent, ki jih lahko potrebujejo drugi žetoni. Izračun predpostavlja, da so vse uteži hkrati v pomnilniku. Nobena od teh številk ni meritev zapakirane izdaje Qwen.

Prenos uteži na CPU ali razporeditev med naprave spremeni mesto shranjevanja bajtov, ne odpravi pa potrebe po njih. Dokumentacija bitsandbytes denimo opisuje nastavitev prenosa na CPU, pri kateri tamkajšnje uteži ostanejo v obliki float32 in se ne pretvorijo v osembitni zapis.[7] To je konkreten razlog za ločena izračuna grafičnega in sistemskega pomnilnika RAM. Oblika zapisa na eni strani takšne razmejitve ni nujno enaka obliki na drugi. Tudi uporaba več naprav zahteva podprt načrt razporeditve; njihove zmogljivosti ne tvorijo samodejno enega poljubno uporabnega pomnilniškega prostora.

Pogovor ima svoje pomnilniške zahteve

Pri ustvarjanju besedila žeton za žetonom lahko transformer ponovno uporabi že izračunane ključe in vrednosti mehanizma pozornosti, namesto da jih računa znova. Predpomnilnik ključ–vrednost oziroma predpomnilnik KV hrani te podatke za prejšnje žetone, in sicer za vsako plast posebej.[2] To je delovni pomnilnik za sklepanje, ne dolgoročna shramba dokumentov ali uporabnikovih nastavitev pomočnika.

Pri običajnem predpomnilniku za celoten kontekst, kjer imajo ključi in vrednosti enake dimenzije, lahko idealno velikost v bajtih izrazimo tako:

```
KV = 2 × L × Hkv × D × T × C × B
```

Faktor dva upošteva ključe in vrednosti; **L** je število plasti, **Hkv** število glav za ključe in vrednosti, **D** dimenzija posamezne glave, **T** število shranjenih žetonov na zahtevo, **C** število sočasnih zahtev in **B** število bajtov na element predpomnilnika. Izračun sledi oblikam tenzorjev ključev in vrednosti po plasteh, ki jih opisuje dokumentacija predpomnilnika Transformers.[2] Če so zahteve različno dolge, izraz **T × C** nadomestimo z vsoto žetonov, ki jih hranimo zanje.

Uporabite število glav za ključe in vrednosti, ne kar števila poizvedbenih glav. Omenjena Qwenova kartica pri pozornosti z združenimi poizvedbami razlikuje med 32 poizvedbenimi glavami in štirimi glavami KV.[3] Če bi v izračun KV vstavili število poizvedbenih glav, bi opisovali drugačen predpomnilnik. Podatke o arhitekturi morate pridobiti za točno izdajo; oznaka »8B« jih ne pove.

Formula ima namenoma omejen obseg. Predpostavlja običajne ključe in vrednosti, shranjene za celoten kontekst, brez skupne rabe predpon, stiskanja, prenosa v drug pomnilnik ali razporeditve med naprave. Transformers opisuje dinamične predpomnilnike, ki rastejo s številom žetonov, predpomnilnike stalne velikosti in predpomnilnike z drsečim oknom, ki starejše žetone izločajo.[2] Ti načini zahtevajo različne ocene dodeljenega pomnilnika. Oglaševani največji kontekst je podprta meja pod določenimi pogoji. Ne dokazuje, da vsaka zahteva rezervira toliko žetonov ali da lahko izbrani računalnik takšno obremenitev vzdržuje.

V tem izračunu sočasnost pomeni zahteve, katerih predpomnilnik mora ostati v pomnilniku, ne števila registriranih uporabnikov. Če aplikacija obdelavo zahteve začasno ustavi, njen predpomnilnik pa obdrži, upoštevajte tudi tega. Prav tako sprememba predpostavljene oblike uteži ne spremeni velikosti elementa predpomnilnika v formuli. Oceno štiribitnih uteži lahko pri načrtovanju združimo z dvobajtnim predpomnilnikom. Za manjši predpomnilnik moramo ločeno spremeniti predpostavke in preveriti, ali izbrano izvajalno okolje podpira želeno obliko zapisa. Manjše porabe predpomnilnika ne smemo pripisati zgolj manjši datoteki uteži.

Izračun za izmišljeno prevajalsko pisarno

Predstavljajte si majhno prevajalsko pisarno, ki pripravlja osnutke prevodov javnih muzejskih opisov. Na enem GPU-ju želi obdelovati štiri sočasne zahteve. Samo za ta primer predpostavimo model z natanko osmimi milijardami parametrov, vsemi shranjenimi v BF16 z dvema bajtoma na parameter, 32 plastmi, osmimi glavami KV, dimenzijo glave 128 in dvema bajtoma na element predpomnilnika. To so učne predpostavke, ne konfiguracija modela Qwen3-30B-A3B.

Posamezna zahteva lahko porabi 6.144 vhodnih žetonov in ustvari največ 2.048 novih žetonov. Pri načrtovanju to pomeni največ 8.192 shranjenih žetonov. V vhodno omejitev vključite navodila, pridobljene odlomke in zgodovino pogovora. Ne namenite celotnega konteksta dokumentu in nato dodajte neomejenega odgovora. Število žetonov določa tokenizator modela; število strani ali besed v dokumentu ni isto kot število žetonov.

Pri navedeni arhitekturi potrebuje vsak shranjeni žeton **2 × 32 × 8 × 128 × 2 = 131,072** bajtov predpomnilnika na zahtevo. Ko to pomnožimo z 8.192 žetoni, dobimo natanko 1 GiB na zahtevo. Štiri takšne zahteve potrebujejo 4 GiB. Velikost samih uteži je **8,000,000,000 × 2 = 16,000,000,000** bajtov oziroma približno 14,90 GiB.

Predpostavimo, da ima aplikacija pred dodeljevanjem svojega pomnilnika na voljo natanko 24 GiB, kar je 25.769.803.776 bajtov. To ni trditev o kartici, ki se prodaja z oznako »24 GB«. Izračun predvideva še 2 GiB za potrebe izvajalnega okolja poleg uteži in podatkov KV ter ločena 2 GiB, ki ju namenoma pustimo neuporabljena kot varnostno rezervo. Nobena od teh dveh postavk ni izmerjena. Sta izrecni začasni predpostavki, ki ju je treba nadomestiti po merjenju, ne splošno veljavna odstotka.

Sočasne zahteve

1

Shranjeni žetoni na zahtevo	8.192
Idealni predpomnilnik KV	1 GiB
Skupaj načrtovani pomnilnik	19,90 GiB

4	
Shranjeni žetoni na zahtevo	8.192
Idealni predpomnilnik KV	4 GiB
Skupaj načrtovani pomnilnik	22,90 GiB

4	
Shranjeni žetoni na zahtevo	16.384
Idealni predpomnilnik KV	8 GiB
Skupaj načrtovani pomnilnik	26,90 GiB

2	
Shranjeni žetoni na zahtevo	16.384
Idealni predpomnilnik KV	4 GiB
Skupaj načrtovani pomnilnik	22,90 GiB

Vsota v vsaki vrstici vključuje isto nezaokroženo velikost uteži, 2 GiB za izvajalno okolje in 2 GiB rezerve; prikazani rezultati so zaokroženi šele na koncu. Načrtovana obremenitev s štirimi zahtevami potrebuje 22,90 GiB, zato poleg izrecne rezerve ostane še približno 1,10 GiB. Če podvojimo vsak kontekst,

predpostavljeno zmogljivost presežemo za približno 2,90 GiB. Če obdržimo daljši kontekst, vendar hkrati dovolimo le dve zahtevi, se izračun vrne na 22,90 GiB.

Ta primerjava pokaže eno od operativnih možnosti. Pisarna lahko daljša opravila postavi v čakalno vrsto, namesto da jih sprejme vsa hkrati. Lahko tudi zmanjša vhod z izbiro dokumentov, omeji dolžino ustvarjenega izhoda ali preveri drugo obliko zapisa modela. Vsaka izbira spremeni storitev: čakalna vrsta podaljša čakanje, izbira dokumentov lahko izpusti koristen kontekst, druga oblika zapisa pa zahteva vrednotenje kakovosti. Načrt naj pove, kakšen kompromis bodo občutili uporabniki.

Zakaj lahko uspešen kratek preizkus zavede

Recimo, da izmišljena pisarna predstavi en kratek pogovor, nato pa omogoči štiri dolge zahteve. Napaka je v tem, da velikost naloženih uteži obravnava kot največjo pomnilniško zahtevo. Pri dinamičnem predpomnilniku potrebe KV rastejo s številom shranjenih žetonov; predpomnilnik stalne velikosti pa lahko svojo zmogljivost rezervira že vnaprej.[2] Predstavitev z drugačnim načinom predpomnjenja ali drugačno obremenitvijo v nobenem primeru ne potrdi predvidene postavitve. Razporejevalnik lahko delo zavrne ali ga postavi v čakalno vrsto, namesto da bi se sistem sesul. Tudi to spremeni, kaj lahko obljubljena storitev dejansko opravi.

Ločiti moramo tudi pomnilnik, ki ga trenutno zasedajo tenzorji, od pomnilnika, ki ga drži upravljalnik dodeljevanja. PyTorch dokumentira upravljalnik s predpomnjenjem, pri katerem se

lahko neuporabljeni predpomnjeni bloki v **nvidia-smi** še vedno prikazujejo kot porabljeni. Razlikuje med pomnilnikom tenzorjev in celotnim pomnilnikom, ki ga upravlja ta mehanizem.**[6]** Rezerviranega in dodeljenega pomnilnika ne seštevajte, kot da gre za neodvisni postavki. Rezervirana vsota že vključuje upravljani pomnilnik, v katerem so tenzorji.

Postavka za izvajalno okolje mora nazadnje pokriti največjo dodatno porabo poleg že upoštevanih uteži in podatkov KV, vključno z začasnimi delovnimi prostori in učinki dodeljevanja. Istih bajtov ne štejte večkrat pod različnimi oznakami. PyTorch pojasnjuje tudi, da **empty_cache()** sprosti neuporabljeni predpomnjeni pomnilnik, ne pomnilnika, ki ga zasedajo živi tenzorji.**[6]** S tem ne moremo rešiti obremenitve, katere nujni podatki presegajo razpoložljivo zmogljivost. Ponavljajoče praznjenje predpomnilnika namesto omejevanja sprejetih zahtev prikriva osnovno težavo.

Dovolj pomnilnika samo po sebi tudi ne pomeni sprejemljive hitrosti. Pri beleženju odzivnih časov ločite začetno obdelavo navodila od poznejšega ustvarjanja žetonov. Konfiguracija, ki paket opravil dokonča čez noč, je lahko primerna za pripravo kataloga in neprimerna za interaktivno delo. Trditev, da konfiguracija zadošča, naj zato pomeni oboje: omejena obremenitev se konča brez napake pomnilnika, izmerjena čakalna doba in čas dokončanja pa ustrezata določeni zahtevi.

Oceno spremenite v ponovljiv preizkus

Pred najemom ali nakupom opreme pripravite zapis preizkusa. Določite revizijo repozitorija modela in zgoščeno vrednost prenesene datoteke, tokenizator, kvantizacijo, različico izvajalnega okolja, gonilnike, napravo ter razporeditev modela. Zabeležite omejitvi konteksta in izhoda, podatkovni tip in način predpomnjenja, omejitev sočasnih zahtev ter uporabo skupnih predpon. Ti podatki pojasnijo, zakaj tuja ugotovitev »gre v pomnilnik« morda ne velja za vaše delo.

Začnite z nalaganjem iz hladnega stanja, nadaljujte z značilno kratko zahtevo, nato pa uporabite najdaljši dovoljeni vhod z najdaljšim dovoljenim odgovorom. Preizkusite načrtovano sočasno obremenitev, ne le zaporednega pošiljanja opravil. Dalj časa trajajočo obremenitev ponovite, da postanejo vidni zadržani pomnilnik in lastnosti čakalne vrste. Vključite zahtevo, ki preseže omejitev, in preverite, ali jo aplikacija jasno zavrne, postavi v čakalno vrsto ali skrajša. Tiho krajšanje lahko ustvari videz uspešnega preizkusa, čeprav odstrani dokaze, potrebne za pravilen odgovor.

Zabeležite največjo porabo pomnilnika naprave, razpoložljive meritve dodeljevanja v izvajalnem okolju, število vhodnih in izhodnih žetonov, čas do prvega izhoda, skupni čas dokončanja, čakanje v vrsti in napake. Ohranite tako ocenjene velikosti tenzorjev kot opažene rezultate. Če je razlika velika, preverite, katera predpostavka ni držala, namesto da nepojasnjeni »faktor dodatne porabe« prilagajate, dokler se številke ne ujemajo. Sprememba sočasnosti, izvedbe predpomnilnika ali kvantizirane datoteke zahteva ponovitev ustreznih primerov.

Za ta članek takšnega strojnega preizkusa nismo izvedli. Izračunani primer pokaže metodo in preveri njeno aritmetiko. Utemeljena odločitev o zmogljivosti ji doda meritve za točno izdajo, izvajalno okolje in obremenitev, priporočilo pa ohrani skupaj z omejitvami, pri katerih je bilo preverjeno.

Viri

Številke virov veljajo za ta vodnik. Povezave vodijo do izvirnih dokumentov. Pred uporabo preverite besedilo, pogoje uporabe in razpoložljivost storitev.

[1] **Hugging Face Transformers: Model memory anatomy**
https://huggingface.co/docs/transformers/main/en/model_memory_anatomy

[2] **Hugging Face Transformers: Caching**
https://huggingface.co/docs/transformers/en/cache_explanation

[3] **Qwen: Qwen3-30B-A3B model card**
https://huggingface.co/Qwen/Qwen3-30B-A3B/raw/main/README.md

[5] **NIST: Prefixes for binary multiples**
https://physics.nist.gov/cuu/Units/binary.html

[6] **PyTorch: CUDA semantics, memory management (upstream source)**
https://raw.githubusercontent.com/pytorch/pytorch/main/docs/source/notes/cuda.md

[7] **Hugging Face Transformers: bitsandbytes quantization**
https://huggingface.co/docs/transformers/en/quantization/bitsandbytes