המאמר היומי של מייק - 06.01.25  
A Survey on Efficient Inference for Large Language Models

המאמר מספק סקירה מקיפה של שיטות לייעול היסק (אינפרנס) ב-LLMs. אז יאללה בואו נסקור את הסקירה.

אתגרים מרכזיים:

1. גודל המודל: מודלי שפה גדולים (ענקיים הכוונה) דורשים משאבי חישוב וזיכרון משמעותיים.

2. סיבוכיות ריבועית (למרות שיש לא מעט שכלולים כמו FlashAttention) של מנגנון ה-attention: מורכבות זו (ביחס לאורך אורך הקלט) משפיעה משמעותית על קצב ההיסק(throughput ו- latency) וצריכת הזיכרון.

3. פענוח אוטורגרסיבי: יצירת טוקנים אחד אחרי השני לא מנצלת באופן מיטבי את משאבי החישוב (כמו GPU) העומדים לרשותנו ופוגעת בתפוקת המודל (throughput)

טקסונומיה של טכניקות אופטימיזציה:

1. אופטימיזציה ברמת הדאטה:

דחיסת קלט: טכניקות כמו חיתוך(pruning) פרומפטים, סיכום(summarization) פרומפטים, דחיסה מבוססת פרומפט רך (למידה של וקטורים רציפים ״המייצגים״ את הפרופמט) , והיסק מבוסס RAG מפחיתות את גודל פרומפטי הקלט תוך שמירה על מידע סמנטי בן. זה יעיל במיוחד לתרחישים הדורשים קלטים ארוכים יותר.

ארגון פלט: שיטות כמו (Skeleton-of-Thought (SoT וגישות מבוססות גרף תלות מאפשרות מקביליות חלקית של גנרוט טוקנים, תוך ניצול המבנה הפנימי של פלטי LLM.

2. אופטימיזציה ברמת המודל:

תכנון מבנה יעיל:

- שיטות כמו (Mixture-of-Experts (MoE מקצים משאבי חישוב באופן דינמי לטוקני קלט, תוך אופטימיזציה של חלקי רשתות MLP הפנימיות בבלוק הטרנספורמר(במימד האמבדינג בד״כ).  
- מנגנוני attention מפושטים או מבוססי-kernel (כמו Performer שסקרתי בזמנו) מפחיתים סיבוכיות מריבועית לליניארית (ביחס לאורך הקלט).  
- חלופות לטרנספורמרים, כמו (State Space Models(SSMs ,כה האהובים עליי, וארכיטקטורות RNN (מתברר שיש פה ושם שימוש בהם) מקטינות את סיבוכיות המודל תוך שמירה על ביצועים תחרותיים (לפעמים). בהקשר זה כדאי להזכיר את Jamba של A21 labs ששילבו ארכיטטקטורת טרנספורמרים עם ממבה (סוג של SSM)

דחיסת מודל:

- קווינטוט: מפחית רוחב סיביות למשקולות והפעלות. שיטות כימות לאחר אימון ואימון-מודע-כימות שומרות על דיוק למרות הדחיסה.  
- דילול: מסיר פרמטרים או ראשי attention מיותרים, באמצעות טכניקות כמו pruning או מנגונני attention דלילים.

- זיקוק ידע(distillation): מאמן מודלים קטנים יותר לחקות את התנהגות המודלים הגדולים, עם אובדן ביצועים מינימלי.

3. אופטימיזציה ברמת המערכת:

שיפורים במנועי היסק (למשל, פענוח ספקולטיבי ואסטרטגיות offloading) ומערכות שירות (למשל, חישוב בבאצים, scheduling מתוחכם וניהול זיכרון) משפרים את ניצול החומרה וביצועי המודל (מבחינת ה-throughput).

המאמר מציין שתהליך ההיסק מחולק לשני שלבים:

1. מילוי מקדים(prefilling): אתחול המודל עם פרומפטי קלט העלאה של זוגות KV שישמשו לגנרוט הטקסט.

2. פענוח: יצירת טוקנים רציפה עם תקורת זיכרון וחישוב.

גישות ניתוח יעילות:

מדדי יעילות כמו השהיה (לטוקן ולרצף כולל), שימוש בזיכרון (משקולות מודל, KV cache, צריכת זיכרון מקסימלית), ותפוקה (טוקנים/שנייה, בקשות/שנייה) מנותחים כדי לכמת את ההשפעה של שיטות אופטימיזציה הנבחנת.

כיוונים עתידיים:

1. טכניקות אדפטיביות המתאימות דינמית את גודל המודל והחישוב בהתבסס על מורכבות הקלט.  
2. אופטימיזציה משותפת בכל הרמות - דאטה, מודל ומערכת - למקסום היעילות.  
3. שיטות מודעות-חומרה לניצול מאיצים מודרניים כמו GPUs ו-TPUs.

https://arxiv.org/abs/2404.14294