⚡️🚀המאמר היומי של מייק 21.08.24: ⚡️🚀

Tree Attention: Topology-Aware Decoding for Long-Context Attention on GPU Clusters

היום נסקור מאמר בנושאה שכבר סקרתי כמה מאמרים לפני כחודש. הנושא הזה נקרא אופטימיזציה והאצה decoding של מודלי שפה כלומר התהליך שגנרוט טוקן חדש בתלות בכל הטוקנים בתוך חלון ההקשר שכבר גונרטו. ואם חלון ההקשר הוא ארוך (מאות אלפי טוקנים) זה יכול לקחת די הרבה זמן בעיקר בגלל מנגנון ה-attention של הטרנספורמרים שמהווים backbone של כל מודלי השפה החזקים.

בשנים האחרונות הוצעו מספר רב של שיטות לייעול והאצה של חישוב ה-attention שהכי מפורסמים מהם הם Flash Attention ו-KV-Cache. שיטות אלו בדרך כלל מנצלות את העובדה שהיום אינפרנס של מודלי שפה מתבצע על GPU וניתן לייעל את החישוב על ידי שימוש ביכולת של GPUs לחשב דברים במקביל.

יתרה מזו מכיוון שמודלי שפה רצים היום על קלסטרים של GPUs יצאו מספר עבודות על איך ניתן לחשב את ה-attention על קלסטרים אלו. מכיוון שמנגנון ה-attention מכיל מכפלות פנימיות (סכומים רבים) אז ניתן לחשבו בצורה מבוזרת די ביעילות.

והמאמר הזה מציע מנגנון מעניין של חישוב ה-attention. הדבר המעניין בו שהמאמר הזה מייצג את חישוב ה-attention (עבור וקטור שאילתה נתון q) כנגזרת של הלוג של ״פונקציה יוצרת״ של ה-attention המחושבת בנקודת 0. פונקציה יוצרת זו נבנית על ידי מניפולציה פשוטה של נוסחת ה-attention וממש מזכירה פונקציה יוצרת של משתנה אקראי.

ניתן להכליל את החישוב הזה ל-attention עבור וקטורי שאילתה q מרובים כאשר במקום נגזרת רגילה יהיה לנו נגזרת לפי n משתנים (n הינו מספר וקטורי השאילתה).

למה זה טוב בכלל? מתברר שהחישוב של attention בצורה כזו מערב פעולות כמו logsumexp ו- max שניתן לבזר אותם בצורה יעילה בין ה-GPUs. החישוב נעשה בצורה של עץ, כלומר מחלקים את הסכומים לכמה חלקים, מחשבים כל חלק ואז מתחילים לסכם את התוצאות בצורה היררכית. זה כמו Map-Reduce רב שלבי.

https://arxiv.org/abs/2408.04093