תכנות מרובה ליבות – תרגול 7#

MESI:

בסוף התרגול הקודם דיברנו על cache. כאשר מממשים מנעול יש צורך להתייחס ל-cache ול-bus, כאשר רוצים להמנע מהעמסה על ה-bus ועל המרחק מהזיכרון עליו עובדים.

NUMA: non uniform memory access:

ארכיטקטורה בה הגישה לזיכרון אינה סימטרית וכאשר מעבד ניגש לזיכרון הפרטי שלו זה נעשה במהירות. נרצה לכתוב אלגוריתמים המתחשבים בכך.

CLH lock:

מנסה לשפר את ה-CLH. במודל זה מחזיקים רשימה של qnodes. ב-CLH מחזיקים רשימה מקושרת בה כל חוט ממתין על ה-node של החוט לפניו. זה יוצר בעיה בארכיטקטורת NUMA כיוון שיתכן מצב בו חוט ייגש לזיכרון מרוחק של חוט אחר כדי לקרוא את הערך שלו. אמנם ההסתובבות עצמה היא על זיכרון לוקאלי, אך אירוע העדכון מגיע מזיכרון מרוחק (או שלא, ניר אמר אחרת בשיעור, כאילו ה-spinning נעשה על הזיכרון המרוחק ולא עותק לוקאלי - הרבה פחות יעיל).

Abortable CLH lock:

הבעיה ב-CLH רגיל היא שחוט לא יכול לצאת באמצע ההמתנה. הפתרון כאן הוא לסמן את ה-node של החוט שרוצה לפרוש כ-aborted, והחוט שאחריו יפסיק להמתין לו וימתין לקודם במקום. כך הממתין הבא ביותר יודע להעביר את ה-spin לחוט האחרון הממתין שלא aborted.

Composite Lock:

ניצור מערך מוגבל בגודל שהוא מערך ה-nodes בעזרתו נסמן את הממתינים למנעול. ננסה לתפוס אובייקט במערך – אם תפסנו מצרפים לרשימה והכל כמו קודם. אם לא הצלחנו לתפוס, מצטרפים ל-node האחרון ברשימה וממתינים יחד עם עוד חוטים, וכאשר זה שלפני יזוז, החוטים שממתינים יחד על אותו Node יתחילו להתפזר.

Aquire:

כאשר חוט רוצה להצטרף הוא מתקן את ה-tail להצביע ל-node שהוא תופס. חוט שמגיע אחרי שהכל תפוס פשוט ממתין על אותו qnode עד שהחוט שממתין עליו ובעל עדיפות לקבל פעם הבאה את המנעול (כשישתחרר לפי האלגוריתם הרגיל) ישחרר את ה-qnode הזה.

היחיד שמחזיר ל-free מ-abort הוא זה שממתין לפניו. כלומר החוט שממתין לחוט שעושה abort, הוא שיסמן את ה-node מ-aborted ל-free ויאפשר לחוט הבא שממתין על אותו qnode שהרגע הפך ל-aborted ואז ל-free לתפוס את אותו qnode.

AtomicStampedReference:

זהו AtomicReference שמאפשר לבצע CAS רגיל אך מותנה ב-expectedStamp מסויים.

חזרה ל-Composite Lock:

Unlock(): ...

TryLock(): בשלב הראשון מנסים להשתלט על node. בשלב השני כשמצליחים מצרפים אותו לרשימה וממתינים לתור שלו – כמו האלגוריתם הרגיל.

- בשלב ה-aquire: מגרילים מקום במערך, ומנסים לעשות עליו CAS. אם לא מצליחים לתפוס, לוקחים את ה-stamp במקום ב-tail ומקבלים את ה-tail הנוכחי והחתימה שלו. כעת ממתינים שיתפנה node. יתכנו כמה מצבים: ה-node שרוצים השתחרר בינתיים, הוא timedout או aborted או released ויש פוטנציאל להשתלט עליו. בודקים האם ה-node האחרון ברשימה הוא זה שרוצים להשתלט עליו, אז מנסים להשתלט עליו ולעשות CAS של ה-tail. אם לא, עושים backoff (המתנה לזמן מסויים קצר), ואם הגענו ל-timeout יוצאים ללא מנעול.
- spliceQNode(): כאשר כן משתלטים על node (הפיכה מ-free ל-wait) מכניסים אותו לרשימה – מוצאים את ה-tail וה-stamp שלו, בודקים שה-timeout לא עבר. אם כן – הופכים אותו חזרה ל-free ויוצאים ב-TimeoutException. אם לא, נכנס לרשימה רק אם tail באותו version של מה שהיה כשתפסנו אותו קודם.

...