Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Párhuzamos programozás

Jámbor Attila (JM38W8), III. évf., BSc mérnök inf. szakos hallgató
Konzulensek: Horváth Ákos PhD hallgató, MIT
Bergmann Gábor PhD hallgató, MIT
Informatikai technológiák szakirány / Rendszertervezés ágazat
Önálló laboratórium 1 összefoglaló
2009/10. II. félév

Az adatfeldolgozás, számítás gyorsításának egyik módja a párhuzamosítás. Párhuzamos feldolgozás során lehetőségünk van az adat vagy a feladat párhuzamosítására. Adatpárhuzamosítás esetén az alkalmazásunk bemenő adatait osztjuk fel több részletre, és ezeken végezzük egy időben a szükséges számításokat. Ezzel szemben az utóbbi esetben pedig a kezdeti feladatunkat bontjuk kisebb részekre, majd ezeket végzik el a különböző számítási erőforrások.

A napjainkban már igen elterjedtek a többmagos processzorral szerelt számítógépek mind a szerverek mind a személyi számítógépek körében. Ezek mellett nem ritkák a többprocesszoros számítógépek, vagy a számítási hálók sem. Annak érdekében, hogy a felsorolt eszközökben rejlő számítási teljesítmény teljes mértékben kiaknázható legyen, alkalmazzák a párhuzamos programozást.

A párhuzamos programozás a szekvenciálistól eltérő gondolkodásmódot követel meg a fejlesztőtől, mely igen sok buktatót rejteget. A fejlesztőnek érdemes ezekkel előzőleg megismerkedni, és felkészülni a nehézségek elkerülésére, illetve kezelésére. Ezek fényében elmondható, a párhuzamos programozás paradigmájának hatékony alkalmazása csak megfelelő tapasztalat birtokában lehetséges, így a félévet azok megszerzésére fordítottam.

Munkám során megvizsgáltam a párhuzamos fejlesztés lehetséges hibaforrásait, úgymint az adatversenyt, mely a helytelen és inkonzisztens adatok lehetséges kiváltó oka, vagy a holtpont jelenségét, melyből (külső vagy belső) segítség nélkül nem tud tovább lépni az alkalmazásunk. Továbbá különféle méréseket is végeztem a párhuzamos feldolgozásból fakadó többletköltség megismerésének érdekében, illetve megvizsgáltam néhány esetlegesen előforduló cache-elési problémát.

Az említett hibalehetőségek hatékony kezelésére különféle módszerek léteznek. Az adatverseny kizárására használtam zárakat illetve olyan objektumtípusokat, melyeken a lehetséges műveletek elvégzése atomikus. A holtpontok kialakulását különböző elkerülési stratégiákkal akadályoztam meg, melyek alkalmazásakor az IBM által fejlesztett, Mtrat nevű eszköz kipróbálása sok segítséget jelentett. A szálkezelésből fakadó többletköltség csökkentésére már léteznek úgynevezett pehelysúlyú szálat biztosító rendszerek. Ezek közül én kettő, a Java Fork-Join Framework és a Kilim használatát ismertem meg. Végül a párhuzamos feldolgozásból eredő cache anomáliák elkerülhetők az általam vizsgált optimalizálási lehetőségek alkalmazásával.

Az ismertetett témakörök bemutatására egyszerű, de szemléletes programokat készítettem Java nyelven.

A félév tapasztalatait összefoglalva elmondható, hogy a párhuzamos feldolgozás nem minden esetben biztosítja a várt sebességnövekedést. Mindemellett alkalmazása rendkívüli körültekintést igényel, mivel a szekvenciális feldolgozás során nem tapasztalható hibákkal szembesülhetünk.