Tisza Ádám
Hd64T1

Ütemező megvalósítása

Dokumentáció

Az ütemező megvalósításakor az első lépés az osztályhierarchia megtervezése volt, főbb osztályok:

- Ütemező (Scheduler)
  - felelősség: létrehozza a termelőket, fogyasztókat, számon tartja a kapott taszkokat, majd feldolgozásra továbbadja egy fogyasztónak.
- Termelő (Producer)
  - felelősség: engedélyezése után folyamatosan új taszkokat készít, és ad át az ütemezőnek.
- Fogyasztó (Consumer)
  - felelősség: az ütemezőtől kapott taszkot lefuttatja, majd jelzi az ütemezőnek, ha végzett.
- Taszk (Task)
  - felelősség: saját azonosítóval, színnel rendelkezik, egy feldolgozandó egységet reprezentál.

Az ütemező az inicializálási fázis után folyamatosan egy ciklusban ad a fogyasztóknak (ha van készenlétben álló) ad a sorokból végrehajtandó taszkokat (ha van ilyen). Az ütemezőben szín szerint külön sorban tárolódnak a taszkok, így könnyen megoldható, hogy egyszerre ne fusson 2 egyszínű taszk, illetve, hogy az azonos színű taszkok érkezési sorrendben kerüljenek feldolgozásra. A nem blokkolt színek egy listában tárolódnak, ha egy adott színű taszkot feldolgozásra odaadunk egy várakozó fogyasztónak, akkor annak a színét ideiglenesen eltávolítjuk ebből a listából (amikor a fogyasztó értesíti az ütemezőt, hogy végzett, akkor újra nem blokkolt lesz a szín), így ilyen színű taszkot nem továbbítunk másik fogyasztónak. A várakozó fogyasztókat hasonló módszerrel kezeli, egy listában tárolja a várakozó fogyasztókat, ha egy fogyasztó taszkot kapott, ideiglenesen eltávolítja a listából, ekkor nem kap új taszkot az ütemezőtől, majd ha a fogyasztó végrehajtotta a taszkot, értesíti az ütemezőt, ekkor újra visszakerül a listába. Mivel minden egyes fogyasztó és termelő külön szálon dolgozik, és ezek a szálak módosítják az ütemező egyes tagváltozóit, ezért minden ezeket módosító függvénynél zárakat alkalmazok, hogy ne zavarhassák egymást a szálak (kivéve a konstruktort, de ekkor még csak egy szál fut).

A termelő(k) létrehozás (és ütemező által engedélyezés) után egy új szálon végtelen ciklusban, adott időközönként létrehoz egy taszkot véletlenszerű színnel, amit továbbad az ütemezőnek, ahol színének megfelelően bekerül egy sorba. A taszk ezután a sorban vár, majd az ütemező átadja egy feldolgozónak. Amikor a feldolgozó elfogad egy taszkot, azt egy új szálon lefuttatja, és a taszk lefutása után értesíti az ütemezőt a

végrehajtáskor, ekkor a taszk bekerül a befejezett taszkok listájába. Arra az esetre, ha valami oknál fogva egy feldolgozó szál meghalna futás közben, van egy figyelő osztály, ami bizonyos időközönként ellenőrzi a feldolgozó szálak állapotát, és ha kiderül, hogy egy szál meghalt, akkor értesíti erről az ütemezőt, az pedig a feldolgozó egység által kezelt taszkot elhelyezi a failedTasks listában, hogy nyomon lehessen követni a hibát. Egy taszknak van egy azonosítója egy színe, és az ellenőrizhetőség miatt egy szín szerinti sorszáma, illetve a taszk osztály tartalmazza a fogyasztó által végrehajtandó függvényt.

Hogy a program futás közben is nyomon követhető legyen, az ütemezőben van egy kezdetleges loggoló rendszer, ami alapján nyomon lehet követni a taszkokkal kapcsolatos fontosabb eseményeket (taszkok létrejötte, feldolgozás kezdete, feldolgozás vége), illetve nagyobb taszk létrehozási/végrehajtási sebességnél hasznosabb csak a teljesítmény loggolása, ekkor csak bizonyos időközönként végrehajtott taszkok számát jegyzi fel az ütemező.

Az átbocsátóképesség függése a színek számától, és a feldolgozóegységektől jól látszik a diagramon: alapvetően több feldolgozóegység esetén nagyobb az átbocsátóképesség, de ha ez meghaladta a színek számát, akkor nem növekszik a hatékonyság, így minél több szín esetén jobb lesz az átbcsátóképesség.