Hibadiagnosztika elosztott rendszerekben

Bevezető

A mérés során megvalósítandó feladatok két csoportba oszthatók: determinisztikus és valószínűségi diagnosztikai algoritmusokra.

- A **determinisztikus algoritmusok** adott feltételek teljesülése esetén képesek teljes és konzisztens diagnosztikai képet előállítani. *Teljesnek* akkor nevezünk egy diagnosztikai eredményt, ha a rendszer minden egysége kapott valamilyen (hibátlan/hibás) minősítést, *konzisztensnek* pedig akkor, ha minden egység valós és az algoritmus által feltárt hibaállapota megegyezik. A teljes és konzisztens diagnosztika eléréséhez teljesítendő további feltétel(eke)re példa a mérési segédletben tárgyalt, a hibás egységek számára adott felső korlát, az ún. t-korlát.
- A **valószínűségi algoritmusok** nem garantálnak biztosan teljes és konzisztens megoldást, de nagy valószínűséggel helyes eredményt szolgáltatnak. A tévedés lehetőségért cserébe ezek a módszerek csak a teszteredményekből kinyert információt használják fel, azaz nem igényelnek a rendszerre nézve olyan további előírásokat, mint a t-korlát a determinisztikus módszereknél.

  A valószínűségi algoritmusok lehetséges diagnosztikai tévedései három csoportba sorolhatók:

  1. Az *ismeretlen egység* egy olyan processzort jelent, amelynek hibaállapotát az algoritmus nem tudta meghatározni. Ebben az esetben a diagnosztika nem teljes.
  2. *Jóindulatú tévedés* esetén egy hibátlan processzort hibásnak minősítünk. Így csökkentjük ugyan a rendelkezésre álló erőforrások számát, de a biztonság javára tévedünk.
  3. *Rosszindulatú tévedés* esetén egy hibás processzort hibátlannak minősítünk. Ekkor a rendszerből nem távolítjuk el a hibás komponenst, ami hibás adatokkal és hibaterjesztéssel az egész folyamat hibáját okozhatja. Az erőforrások számát nem csökkentjük.

A mérési feladat

A mérési feladat a következő pontban leírt valószínűségi diagnosztikai algoritmus megvalósítása egy C nyelven megírt diagnosztikai szubrutin formájában. A szubrutint a szimulációs környezetbe ágyazva le kell fordítani, majd az algoritmus helyességét próbafuttatásokkal ellenőrizni. (A futtatások eredményét statisztikailag értékelve a mérési jegyzőkönyvben is fel kell tüntetni!)

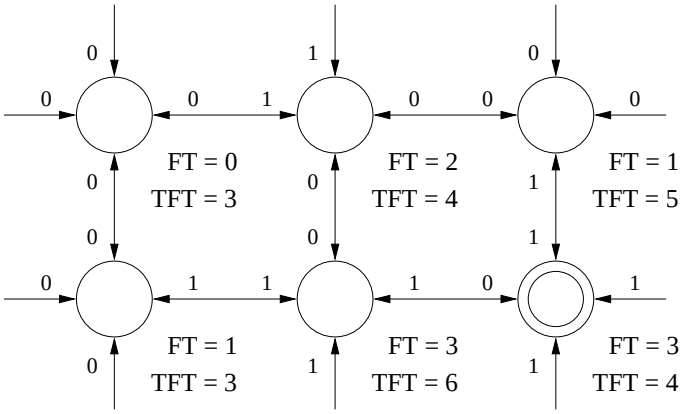
A mérés kiértékelése

A valószínűségi algoritmusok „tévedhetnek”. A jó- vagy rosszindulatú tévedések száma az adott valószínűségi algoritmus fontos jellemzője. Ezért a valószínűségi módszerek kiértékelésekor az algoritmus értékeléséhez több pontban elvégzett mérések alapján (adott rendszerméret esetén növekvő hibaszám, vagy adott számú hiba mellett növekvő rendszerméret, stb.) adjon statisztikát (táblázatos és diagram alakban) a jóindulatú és rosszindulatú diagnosztikai tévedések (és ha van, az ismeretlen egységek) számáról!

Szorgalmi feladat: ha tud, adjon ötleteket az adott algoritmus továbbfejlesztéséhez (és ha van idő, valósítsa is meg)! Cél, hogy a diagnosztikai tévedések száma csökkenjen.

A megvalósítandó algoritmus leírása

Dahbura és társai a következő egyszerű valószínűségi diagnosztikai algoritmust dolgozták ki: a kétdimenziós tórusz topológiájú hálózatban minden processzor teszteli mind a négy szomszédját. Ezek után minden processzornál összesítjük azokat a hibás teszteket, amelyek rá vonatkoznak. Erre mutat példát a 1. ábra.



1. ábra. Az FT és TFT értékek számítása

Ez az összeg jellemzi, hogy mennyire tartható hibásnak egy adott processzor. Amennyiben ez a szám meghalad egy előre meghatározott limitet, akkor a processzort rossznak minősíthetnénk, ha nem, akkor jónak. Ez a durva módszer (ami egyébként megfelel a mérési segédletben szereplő példaprogramnak) azonban sok diagnosztikai tévedéshez vezetne, hiszen a tesztelők között lehetnek rosszak is. Ezért figyelembe kell venni a tesztelők „megbízhatóságát", azaz a tesztelőkre vonatkozó hibás teszteket is.

Dahbura algoritmusa ezért minden processzorra megszámolja, hogy a szomszédai közül hányan tesztelték őt hibásnak (FT) és hányan tesztelték hibásnak az ő tesztelőit (TFT). Miután minden egységre kiszámította ezt a két számértéket, kiválasztja a legnagyobb FT és legkisebb TFT értékkel rendelkező processzort és azt hibásnak minősíti.

Ezután az éppen hibásnak minősített processzort és általa végzett tesztek eredményét „kiveszi a rendszerből", azaz hibátlanra változtatja. Ekkor megint újraszámolja a megmaradt processzorokra az FT és TFT értékeket (valójában csak csökkenteni kell a megfelelő számlálókat), és megint megkeresi az új helyzetben legnagyobb FT és legkisebb TFT értékkel rendelkező processzort. Ezt is hibásnak minősíti és a teszteredményeit ennek is kiveszi a rendszerből.

Az iteráció addig tart, amíg van a rendszerben hibás teszt. Az eljárás befejezése után minősítés nélkül maradó processzorok „hibátlan" címkét kapnak.
Valósítsa meg Dahbura algoritmusát!