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 determinisztikus 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 determinisztikus algoritmusok az adott topológiára érvényes t-korlát értékével megegyező számú hibás egységig helyesen (teljes és/vagy konzisztens módon) *kell* működniük. A t-korlát felett azonban már nincs erre garancia. Ezért a determinisztikus algoritmusok ellenőrzésekor:

1. Több pontban elvégzett mérések alapján (adott rendszerméret esetén növekvő hibaszám, vagy adott számú hiba mellett növekvő rendszerméret, stb.) igazolnia kell, hogy a t-korlátig az algoritmus nem téved!

2. Ha ez teljesült, adjon statisztikát (táblázatos és diagram alakban) a t-korlát feletti hibaszámokra a jóindulatú és rosszindulatú diagnosztikai tévedések (és ha van, az ismeretlen egységek) alakulásáról.

Szorgalmi feladat: ha tud, adjon ötleteket az adott algoritmus továbbfejlesztéséhez (ha van ideje, valósítsa is meg)! Cél, hogy a diagnosztikai hatékonyság növekedjen (ilyen lehetőség pl. a felfedezett ellentmondások alapján meghozható biztos következtetések felhasználása).

A megvalósítandó algoritmus leírása

Hakimi és Nakajima *adaptív* diagnosztikai módszert adtak PMC rendszerekhez. Az adaptív algoritmusok jellegzetessége az, hogy a tesztkészletet a futásuk során a már elvégzett tesztek eredményeitől függően választják meg. Ennek eredményeképp átlagosan sokkal kevesebb tesztet igényelnek, mint a nem adaptív módszerek.
A Hakimi, Nakajima algoritmus a következő két megállapításon alapul:

- ha a teszteredmény $a_{ij} = 0$, akkor nem lehetséges, hogy u_i hibátlan miközben u_j hibás, és
- ha a teszteredmény $a_{ij} = 1$, akkor nem lehetséges, hogy u_i és u_j mindketten hibátlanok.

Az algoritmus alapgondolata az, hogy elég egyetlen biztosan hibátlan egységet megtalálni, mert azután a hibátlan processzorból kiindulva és mindig hibátlannak tesztelt egységeken végiglépkedve a teljes rendszer hibaállapota felderíthető.

A hibátlan egység megtalálásához a következő megközelítés alkalmazható. Készítsünk két listát:

- a valószínűleg hibátlan egységeket tartalmazó U_h, és
- a gyanús egységeket tartalmazó U_d listákat.

Kezdetben mindkét lista üres. Vegyünk egy tetszőleges u_i egységet és tegyük be az U_h listába. Ezek után a következőt végezzük egy ciklusban:

1. Amíg $|U_h| + |U_d|/2 \leq t$, addig kivesszük az U_h lista utolsó elemét (legyen ez u_j) és teszteltessük őt az egyik u_k szomszédjával.
2. Ha a teszteredmény $t_{k,j} = 0$, akkor tegyük vissza u_j-t és utána u_k-t is az U_h lista végére. (Így a következő lépésben már u_k-t vesszük ki és teszteltetjük egy eddig nem szerepelt szomszédjával.)
3. Ellenkező esetben, ha $t_{k,j} = 1$, azaz a teszt hibát jelzett, akkor mind az u_j, mind az u_k egységet áttesszük az U_h listából az U_d listába.

Ha a fenti lépések során elérjük a $|U_h| + |U_d|/2 = t + 1$ határt, akkor megállhatunk: az U_h lista legelső eleme *biztosan hibátlan* egység. Ezután, ahogy említettük, a megtalált biztosan hibátlan egységtől kiindulva és a hibátlannak tesztelt egységeken végiglépkedve a teljes rendszer hibaállapota felderíthető. (Ha nem sikerül az első megtalált biztosan hibátlan egység segítségével a teljes rendszert minősíteni, akkor a megmaradó, még nem diagnosztizált egységek közt folytatni kell a módszert elölről.)
Valósítsa meg a Hakimi, Nakajima algoritmust!