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

Kameda, Toida és Allan egy olyan algoritmust adott a t-korlátos diagnosztika problémájára, amely több későbbi módszerhez szolgált alapötletül:

1. Az algoritmus első lépésben kiválaszt egy tetszőleges u_i egységet, és feltételezi, hogy az hibátlan.

2. A további lépések során minden u_i egységhez két halmazt állít elő: az $L^0(u_i)$ jelű *hibátlannak adódó* egységek halmazát és a $L^1(u_i)$ jelű *hibásnak adódó* egységek halmazát. A két halmaz előállításában a következő szabályokat használja:

Egység	**Teszteredmény**	**Következtetés**
$u_j \in L^0(u_i)$	$t_{j,k} = 0$	$L^0(u_i) \leftarrow L^0(u_i) \cup u_k$
$u_j \in L^0(u_i)$	$t_{j,k} = 1$	$L^1(u_i) \leftarrow L^1(u_i) \cup u_k$
$u_j \in L^0(u_i)$	$t_{k,j} = 1$	$L^1(u_i) \leftarrow L^1(u_i) \cup u_k$
$u_j \in L^1(u_i)$	$t_{k,j} = 0$	$L^1(u_i) \leftarrow L^1(u_i) \cup u_k$

3. Amennyiben a halmazok előállítása során talál olyan egységet, amely mindkét halmaznak része, tehát $L^0(u_i) \cap L^1(u_i) \neq \emptyset$, akkor az algoritmus *ellentmondásra jutott*. Mivel az ellentmondás a u_i hibátlan feltevésből származott, ezért u_i biztosan hibás. Ez egy ún. „erős kötés". Ekkor u_i bekerül a hibásnak talált egységeket tároló U_f halmazba. Az U_f halmazt az u_i egység hibás állapotából levonható következtetésekkel bővítjük. Az $L^0(u_i)$ és a $L^1(u_i)$ halmazokat töröljük. Végül új u_i egységet választ és újraindul az algoritmus.

4. Ha már nem tudjuk az $L^0(u_i)$ vagy az $L^1(u_i)$ halmazokat tovább bővíteni és $|U_f \bigcup_{\forall i} L^1(u_i)| \leq t$, akkor u_i hibátlan állapota konzisztens a levonható következtetésekkel és a t-korláttal. Ekkor a $L^0(u_i)$ halmazban levő egységek „hibátlan", a $L^1(u_i)$ halmazban levők „hibás" címkét kapnak. Az ilyen címkézéseket „gyenge kötésnek" nevezik. Ha a hibás egységek száma nem haladja meg a t korlátot, de u_i hibás, akkor a $L^1(u_i)$ halmazban levő egységek „hibás" címkét kapnak.

5. Ha $|U_f \bigcup_{\forall i} L^1(u_i)| > t$, akkor az algoritmusnak vissza kell lépnie. Ilyenkor a megelőző lépésben kiadott gyenge kötéseket vissza kell vonni, az akkor vizsgált u_j egység címkéjét az ellenkezőre kell változtatni, és újra kiszámolni a két következményhalmazt. (Ehhez a gyenge kötéseket egy LIFO stack jellegű struktúrában érdemes tárolni). A diagnosztika elkészült, ha a t-korláttal kompatibilis, minden egységre kiterjedő diagnosztikát sikerül létrehozni.

Ha az algoritmus vissza akar lépni, de a gyenge kötések stackje üres, akkor diagnosztika nem képezhető, mert a rendszer nem tartja be az egy lépésben t-hiba diagnosztizálhatóság feltételeit.
Valósítsa meg a KTA algoritmust!