

Hibatűrés

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

<http://www.mit.bme.hu/>

A redundancia tipikus alkalmazása

- **Hardver tervezési hibák (< 1%):**
 - Tipikusan nem számítanak rá (jól tesztelt komponensek)
 - Eltérő tervezésű hardver lenne szükséges
- **Hardver állandósult működési hibák (~10%):**
 - Hardver redundancia (pl. tartalék processzor)
- **Hardver időleges működési hibák (~70-80%):**
 - Szoftver redundancia (pl. állapotmentés és helyreállítás)
 - Idő redundancia (pl. utasítás újravégrehajtás)
 - Információ redundancia (pl. hibajavító kódolás)
- **Szoftver tervezési hibák (~20-30%):**
 - Szoftver redundancia (pl. eltérő tervezésű modulok)

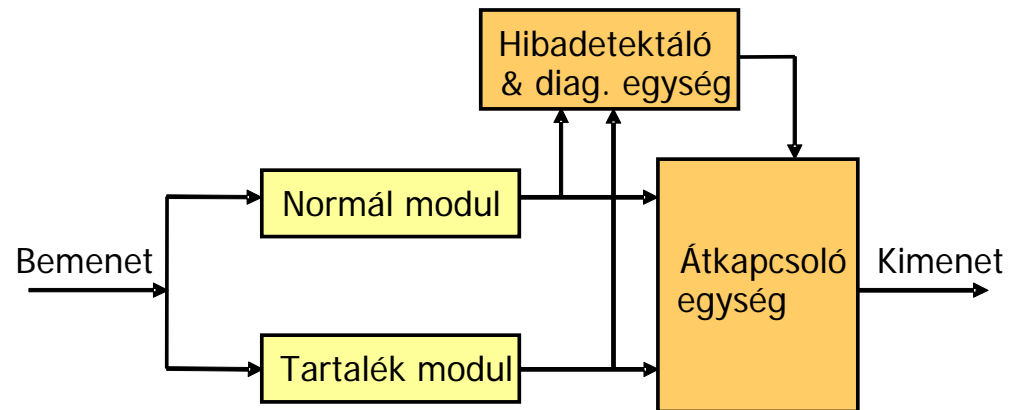
Hibatűrés állandósult hardver hibák esetén

Hardver redundancia

Többszörözés:

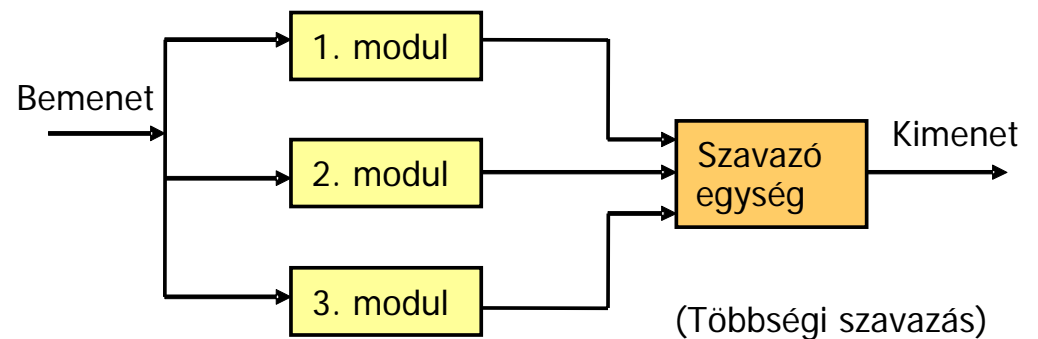
- Kettőzés:

- Hibadetektálás összehasonlítással
- Hibatűrés: Diagnosztikai támogatás és átkapcsolás



- TMR: Triple-modular redundancy

- Hiba maszkolása többségi szavazással
- Szavazó kritikus elem (de egyszerű)



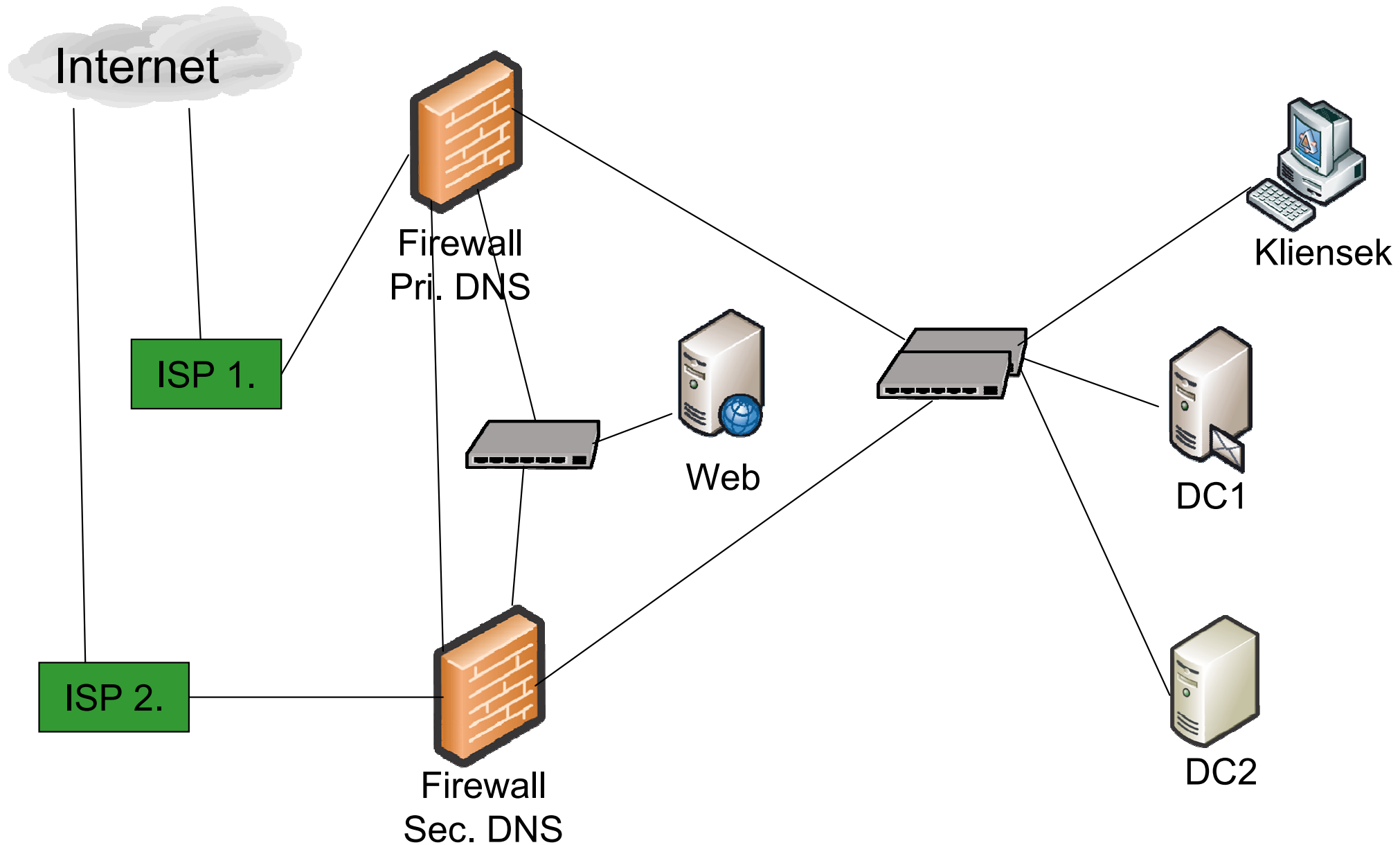
- NMR: N-modular redundancy

- Hiba maszkolása többségi szavazással
- Missziós idő túlélése nagyobb esélyű, utána javítás jöhet
- Repülőgép fedélzeti eszközök: 4MR, 5MR

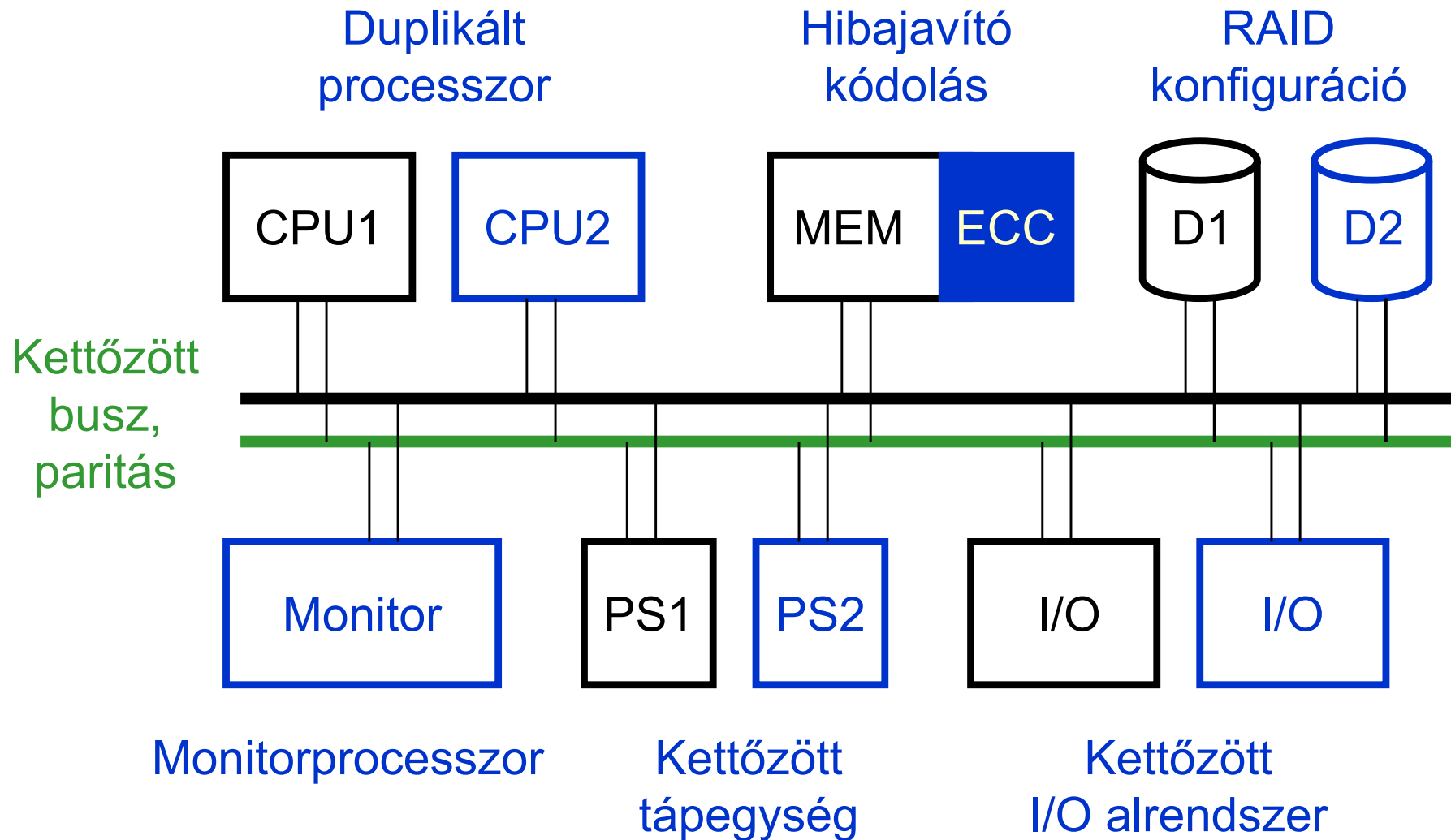
A redundancia szintje

- **Komponens** (szerver) szint: Lazán csatolt
 - Nagy rendelkezésre állású szerver fürtök (feladatátvételi fürtök)
pl. HA Linux
 - Szoftver támogatás: állapotszinkronizálás, feladatátvétel
- **Kártya** szint:
 - Futásidőbeli átkonfigurálás “hot-swap”
pl. compactPCI, HDD
 - Szoftver támogatás: konfigurációkezelés
- **Alkatrész** szint: Szorosan csatolt
 - Alkatrész szintű többszörözés
pl. TMR, önellenőrző áramkörök

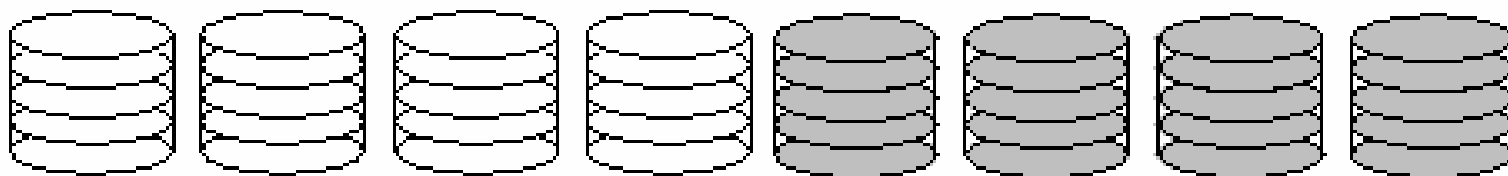
Példa: Komponens szintű redundancia...



Példa: Kártya szintű redundancia...



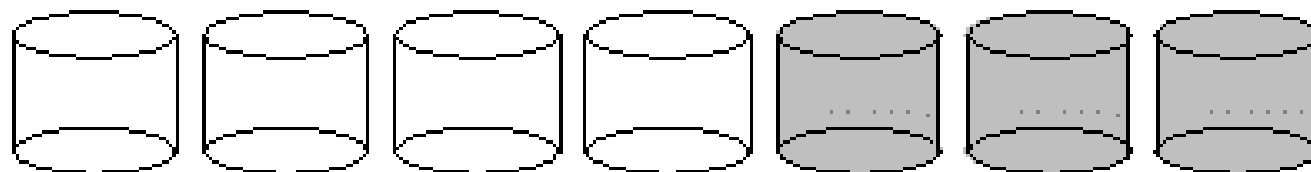
Tükrözött
diszkek



Hibajavító
kódolás

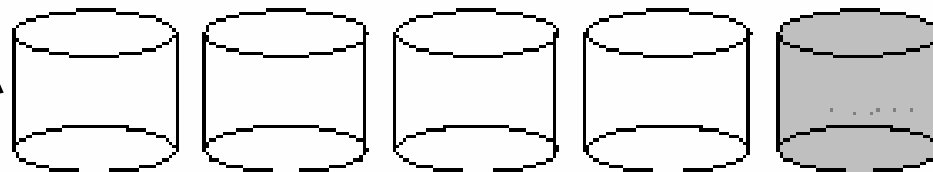
RAID-1

Azonosítható
a hibás diszk:
Paritás elég
a javításhoz



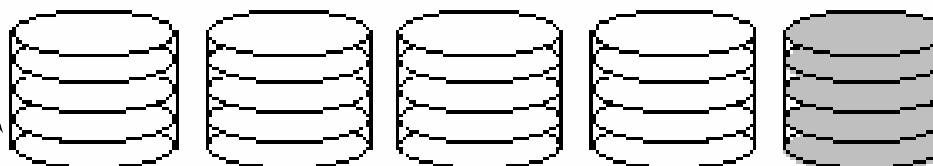
RAID-2

Konkurens
hozzáférés a
blokkokhoz

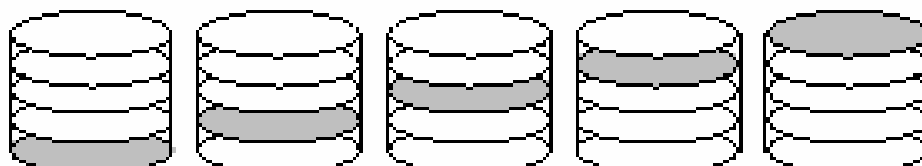


RAID-3

Paritásdiszk
sem szűk
kereszt-
metszet



RAID-4



RAID-5

További RAID konfigurációk

- RAID-6
 - Kétféle elosztott paritás
 - Két diszk kiesése is tolerálható
 - Helyreállítás alatt is van redundancia
- RAID 0+1
 - Kombinált: Teljesítmény és hibatűrés ötvözése
 - Diszkek összefűzve (RAID-0), majd ez tükrözve (RAID-1)
 - Egy hiba esetén a tükrözés megszűnik
- RAID 1+0
 - Diszkek tükrözve (RAID-1), majd ez összefűzve (RAID-0)
- RAID 5+0
 - RAID-5 összefűzve
- RAID 5+1
 - RAID-5 tükrözve

Hibatűrés időleges hardver hibák esetén

Időleges hardver hibák kezelése

Megközelítés: Szoftver alapú

- Ismételt végrehajtás esetén a hiba nem jelentkezik
- Hibahatások kiküszöbölése a fontos

A hiba kezelhető **hibamentes állapot beállításával**
(és ismételt végrehajtással)

Feladatok (fázisok):

1. Hibadetektálás
2. Hibahatás felmérése
3. Helyreállítás
4. Hibaok (meghibásodás) kezelése

1. Hibadetektálás

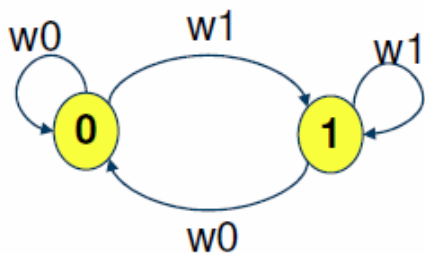
- **Alkalmazásfüggetlen** mechanizmus:
PI. processzor, MMU, operációs rendszer szintjén
 - Illegális utasítás
 - Védelmi szintek, jogosultságok (pl. memória hozzáférés)
- **Alkalmazásfüggő** ad-hoc módszer:
 - Időzítések ellenőrzése
 - Hihetőségvizsgálat
 - Visszahelyettesítés (algoritmus)
 - Struktúra ellenőrzés
 - Diagnosztikai ellenőrzés
 - ...

Példa: SAFEDMI

- Indításkor:
 - Részletes öntesztelés
 - Processzor („core” utasítások alapján) + watchdog timer
 - Memória: teszt algoritmusok
- Működés közben:
 - Periodikus tesztelés
 - Kisebb erőforrásigényű technikákkal
 - On-line ellenőrzések:
 - Kommunikáció, konfigurálás: Hibadetektáló kódolás, adatok elfogadhatósági ellenőrzése
 - Vezérlési funkciók: Vezérlési gráf ellenőrzése (jelzőszámok)
 - Intenzív adatfeldolgozás: Duplikált végrehajtás és összehasonlítás

Példa: Memória tesztelése

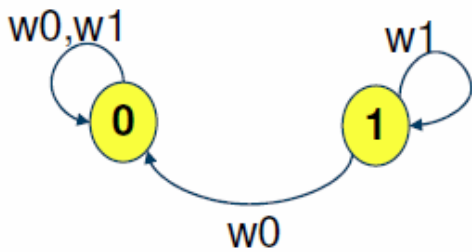
Hibamentes cella:



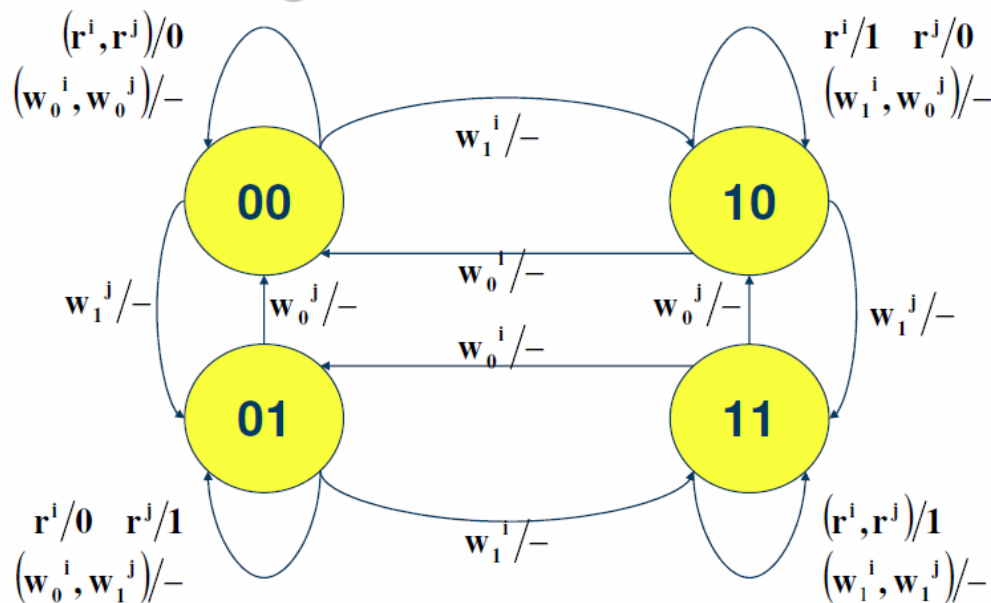
Leragadási hibák:



Tranzíciós hibák:



Összeragadás tesztjéhez:



„March” algoritmusok:

				1
			1	
		1		
	1			
1				

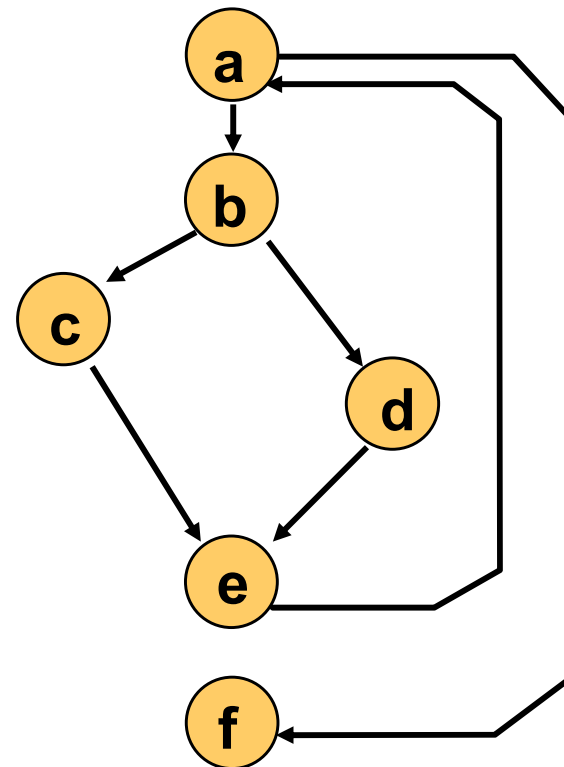
Példa: Szoftver önellenőrzés

- Végrehajtási útvonalak ellenőrzése
 - Referencia: Vezérlési gráf alapján
 - Aktuális futás: **Jelzőszámok** alapján ellenőrizhető

Forráskód:

```
a: S(a); for (i=0; i<MAX; i++) {  
b:   S(b); if (i==a) {  
c:     S(c); n=n-i;  
      } else {  
d:     S(d); m=m-i;  
      }  
e:   S(e); printf(“%d\n”,n);  
      }  
f:   S(f); printf(“Ready.”)
```

Vezérlési gráf:



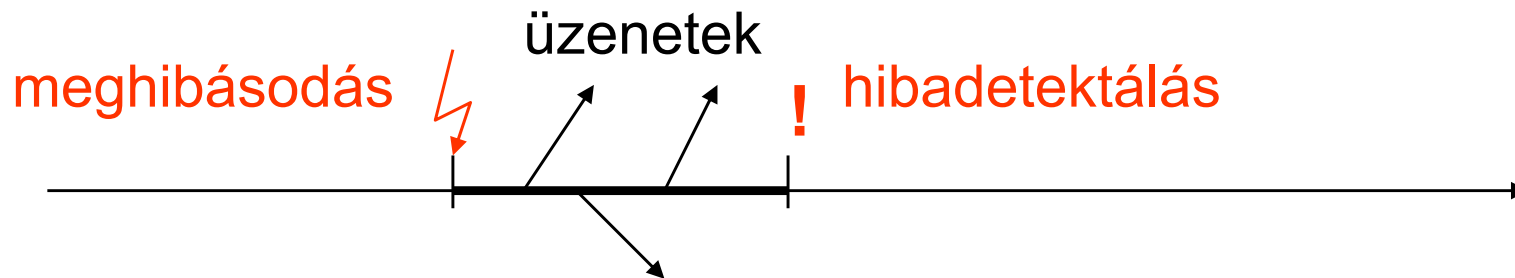
Járulékos feladatok

Keretrendszer hibadetektáláshoz (monitorozáshoz)

- Alkalmazásfüggő technikák beillesztése
- Detektálás ütemezése
- Hibajelzések kiadása
 - Ismételt hibajelzések kiszűrése
 - Függőségek kezelése (root cause analysis)
 - Naplózás
- Megszűnő hiba jelzése
- Hozzáférés hibajelzésekhez
 - Nézetek, szerepek
 - Nyugtázás

2. Hibahatások felmérése (kárfelmérés)

- Hibadetektálás késleltetési ideje alatt a hiba terjed
 - Pl. elosztott rendszer komponensei között, processzek között



- Hibaterjedés **behatárolása**
 - Interface (kimeneti, bemeneti) ellenőrzés
 - Erőforrásokhoz való hozzáférés korlátozása
 - Atomi jellegű műveletek kialakítása
- Hibaterjedés **felmérése**: interakciók követése
 - Interakciók naplózása
 - Diagnosztikai ellenőrzések

3. Helyreállítás

Egyszerű technikák:

- Retry, restart, reboot; meleg reset, hideg reset; frame boundary

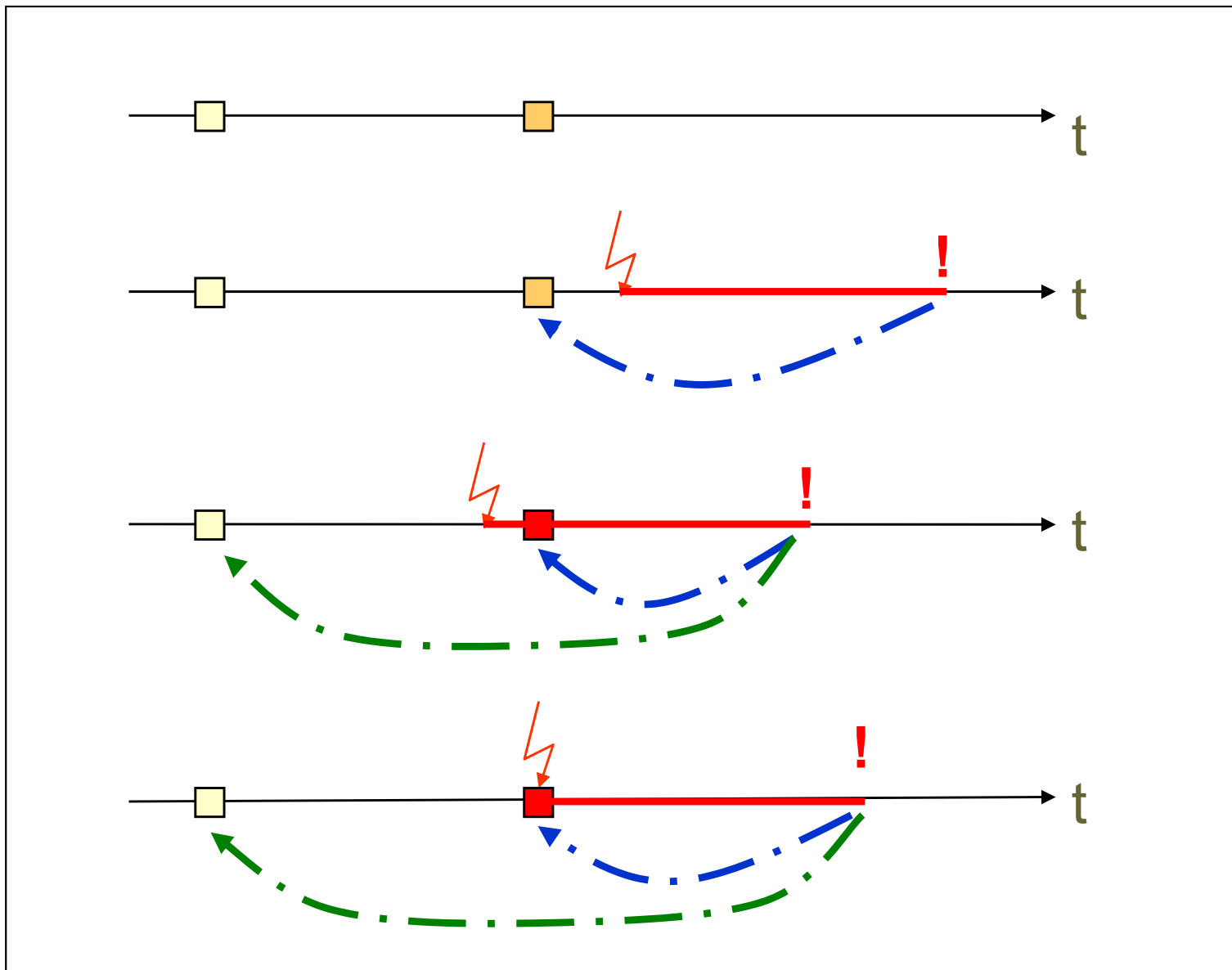
Összetett technikák:

- **Előrelépő helyreállítás:**
 - Hibamentes állapot beállítása: **szelektív korrekció**
 - A detektált hiba és a hibahatás függvénye
 - Előre figyelembe vett hibák esetén
- **Visszalépő helyreállítás:**
 - **Előző hibamentes állapot beállítása**
 - Hibától függetlenül megvalósítható
 - Állapotmentés és visszaállítás minden komponensre
- **Kompenzáció:**
 - Többlet információ alapján a hibahatás kompenzálható

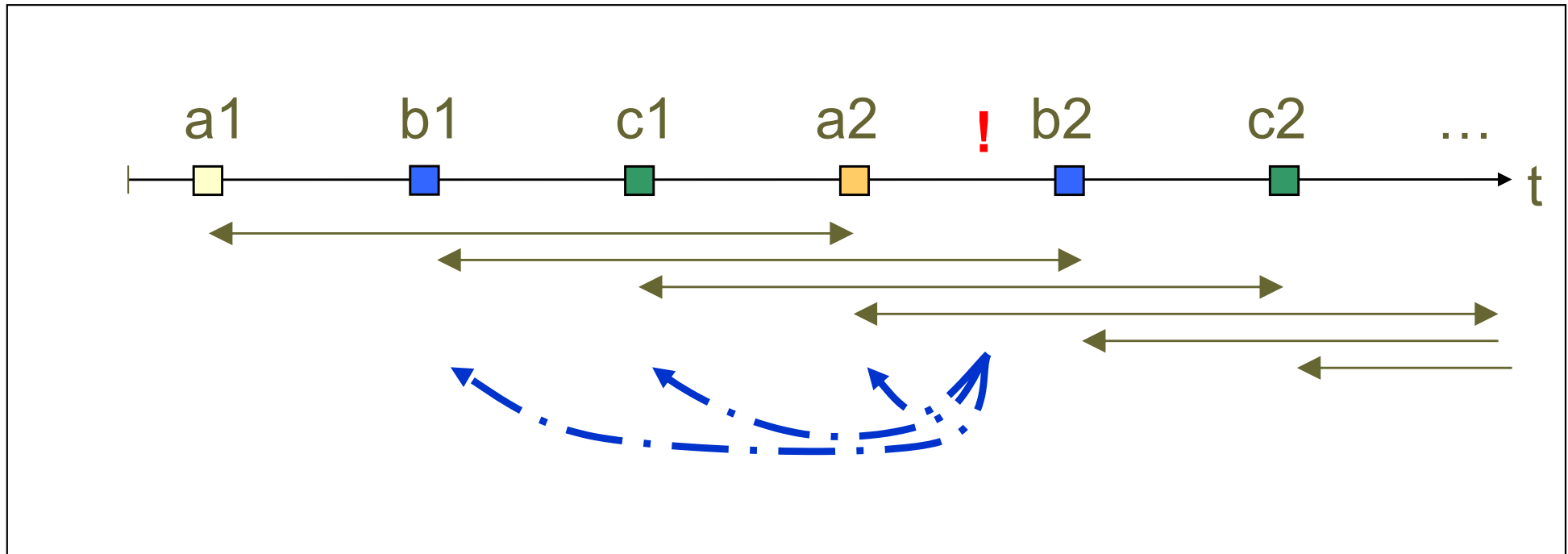
Visszalépő helyreállítás

- **Állapotmentés** alapján
 - Checkpoint: állapotmentés (időpontja)
 - Műveletek:
 - **Állapotmentés**: időközönként, üzenetek után
 - **Visszaállítás**: mentésből az operatív memóriába
 - **Eldobás**: adott számú checkpoint megtartása
 - Feltételezett hiba: időleges hardver hiba
 - Pl: “autosave”
- **Műveletek visszavonása** alapján
 - Műveletek **naplózása** szükséges
 - Feltételezett hiba: a téves vagy szándékolatlan művelet
 - Pl. többszintű “undo”
- **Kombinált módszer**

Visszalépő helyreállítás forgatókönyvei



Checkpoint tartományok

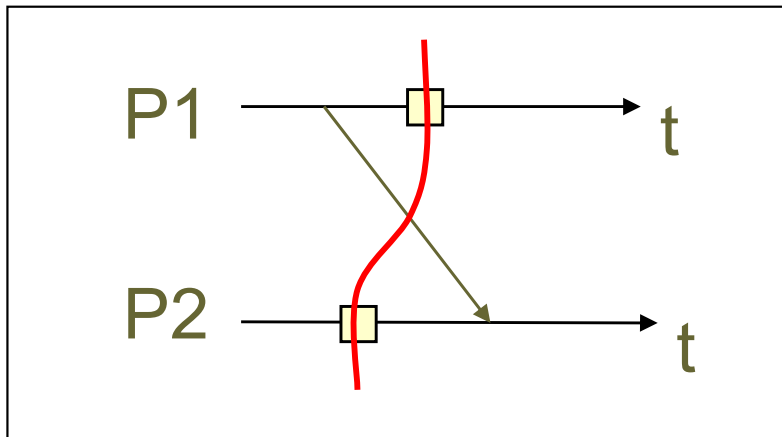


Optimalizálás szempontjai:

- Korlátos méretű tár az állapotmentéshez
- Visszalépéssel újravégzendő (elvesztett) számítások
- Hibadetektálás lappangási ideje

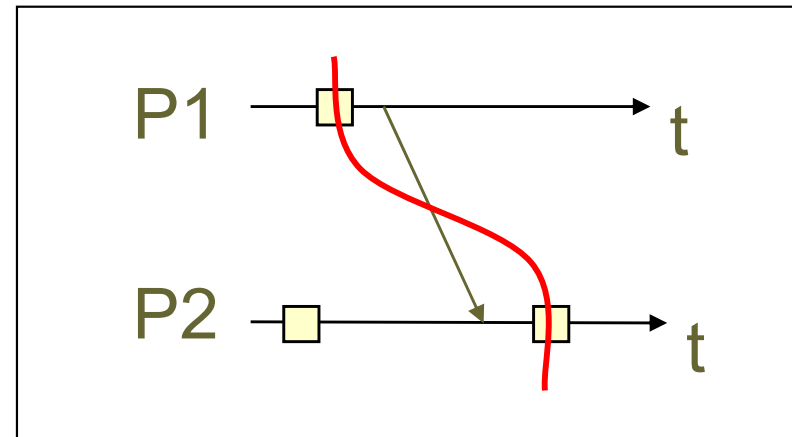
Helyreállítás elosztott rendszerekben

Üzenetek „metszik” a helyreállított állapotok által kijelölt vágatot:



Függő üzenet:

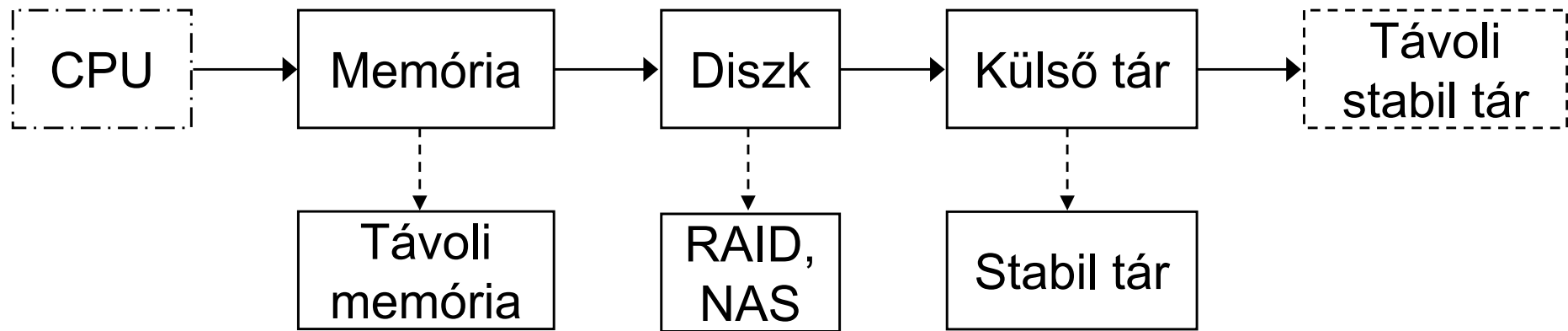
- Előtte elküldött
- Utána feldolgozott



Inkonzisztens üzenet:

- Utána elküldött
- Előtte feldolgozott

Állapotmentések visszalépő helyreállításához



- Jellegzetes használati esetek:
 - Hosszú számítási idejű alkalmazások
 - Tudományos számítások, folyamatos működésű alkalmazások
 - „Commodity computing”: Skálázhatóság olcsó hardverekkel
 - Adat analízis alkalmazások: Pl. Google: Átlagban 5 számítógép kiesése egy-egy Map-Reduce job végrehajtása alatt
 - Nagyméretű infrastruktúrák: 4000 számítógépből álló hálózat: 6 óránként egy diszk kiesése

Kihívások

- Komponensek visszaléptethetősége
 - Pl. CPU állapot mentése és visszatöltése
 - Determinisztikus működés
- Környezettel való együttműködés kezelése
 - Bemenet: naplózás
 - Kimenet: késleltetés vagy ellensúlyozás
- Megbízhatóság és helyreállítási garancia
 - Hibadetektálás hibafedésének növelése, késleltetési idejének csökkentése
 - Mentés helyének megbízhatósága (replikák)
- Hatékonyság növelése
 - Konkurens állapotmentés
 - Másolatok használata (shadow update, recovery cache)
 - Többszintű állapotmentés („soft” és „hard” checkpoint)

Mentések hatékonysága

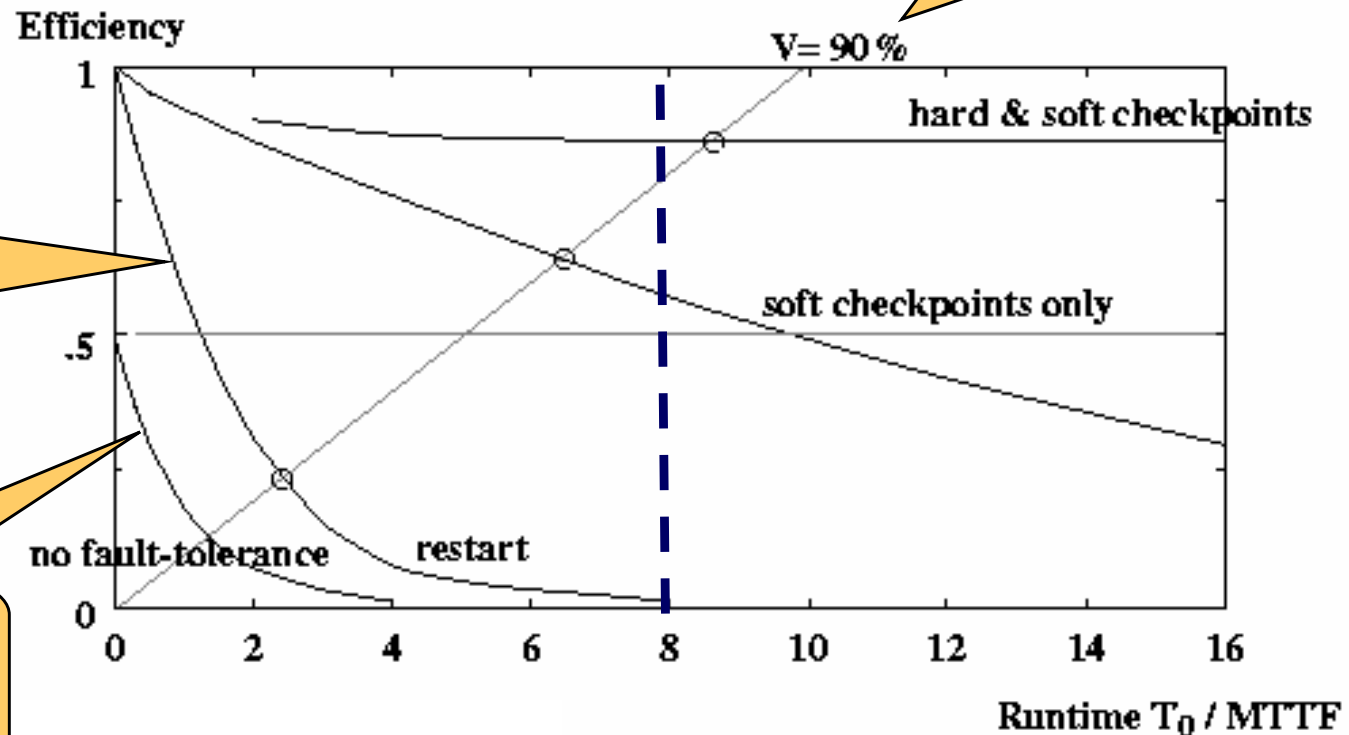
Itt a hatékonyság:

$$\frac{\text{referencia (hibamentes) futási idő}}{\text{hibakezeléssel mért futási idő}}$$

V=90% vonal:
A hibadetektálás 99%
hibafedése esetén 90%-os
bizalom az eredmények
helyességében a vonal feletti
területen

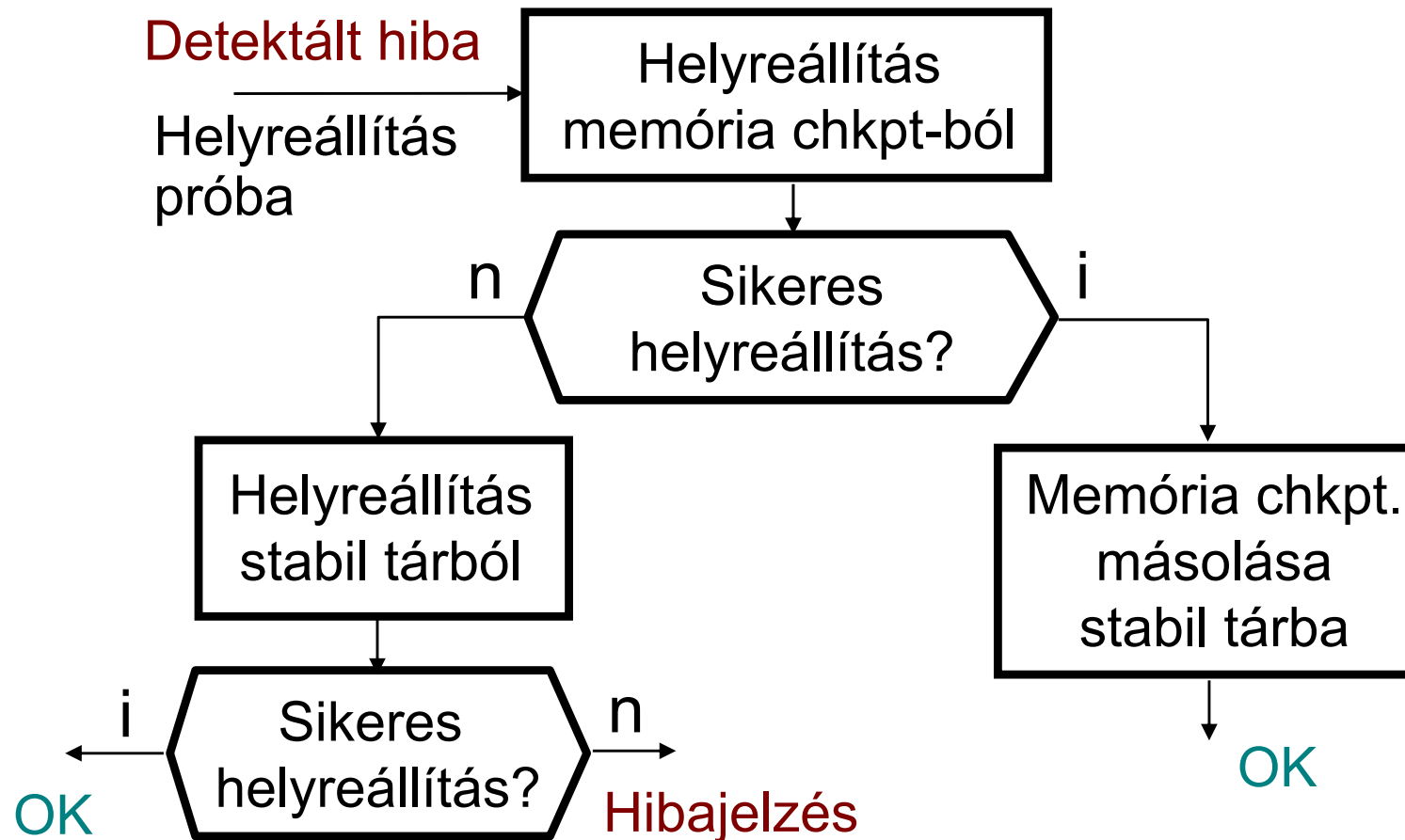
99% hibafedésű
hibadetektálás van

Újra és újra futtatás,
míg ugyanaz az
eredmény ki nem jön



Többszintű állapotmentés

- Alapelv:
 - Gyakori mentés gyors memóriába (memória chkpt.)
 - Ritka mentés perzisztens tárba (stabil tár)



Stabil tár koncepciója

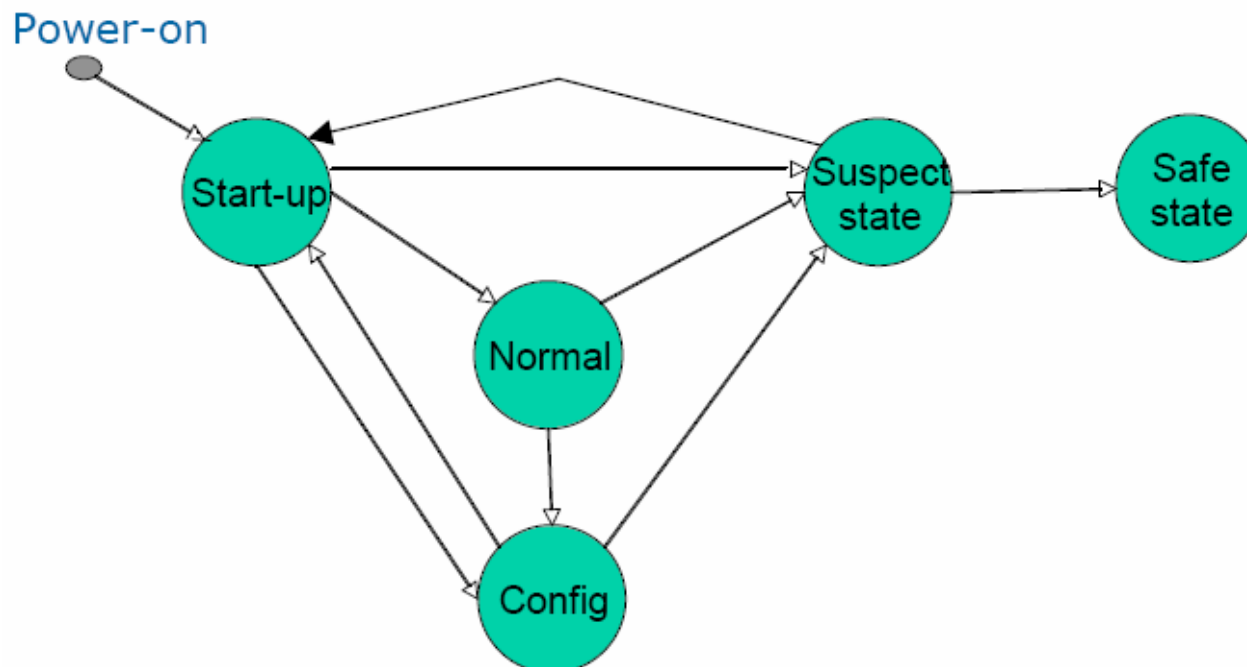
- Többféle megvalósítás lehetséges
- Elvárt tulajdonságok
 - Perzisztens
 - Atomi hozzáférésű
 - Autonóm működésű
 - Hibajavítást alkalmazó
 - Információrejtést alkalmazó

4. Meghibásodás (hibaok) kezelése

- Időleges hibák:
 - Előre- vagy visszalépő helyreállítás elég
- Állandósult hibák:
 - Helyreállítás nem lesz sikeres (újra detektált hiba)
 - Ez mechanizmus lehet az állandósult hiba felismerésére
 - Hiba lokalizálás (diagnosztika) után beavatkozás kell:
 - Újrakonfigurálás:
Hibás komponens feladatainak átvétele
 - Javítás, csere
 - Graceful degradation:
Csökkentett funkcionalitású működés,
de a kritikus funkciókat megtartva

Példa: SAFEDMI hibakezelés

- Működési módok:
 - Startup, Normal, Configuration
- Hibadetektálás esetén: Tranziens hiba feltételezése
 - Visszalépés **Startup állapotba** és részletes önteszt
- Állandósult hiba lehetőségének kezelése:
 - Visszalépés **Suspect állapoton** keresztül:
Hibaszámlálás (adott idő alatt korlátozott számú visszalépés megengedett)
 - A hibaszám túllépése esetén **Safe állapotba** lépés (leállítás)



Hibatűrés szoftver tervezési hibák esetén

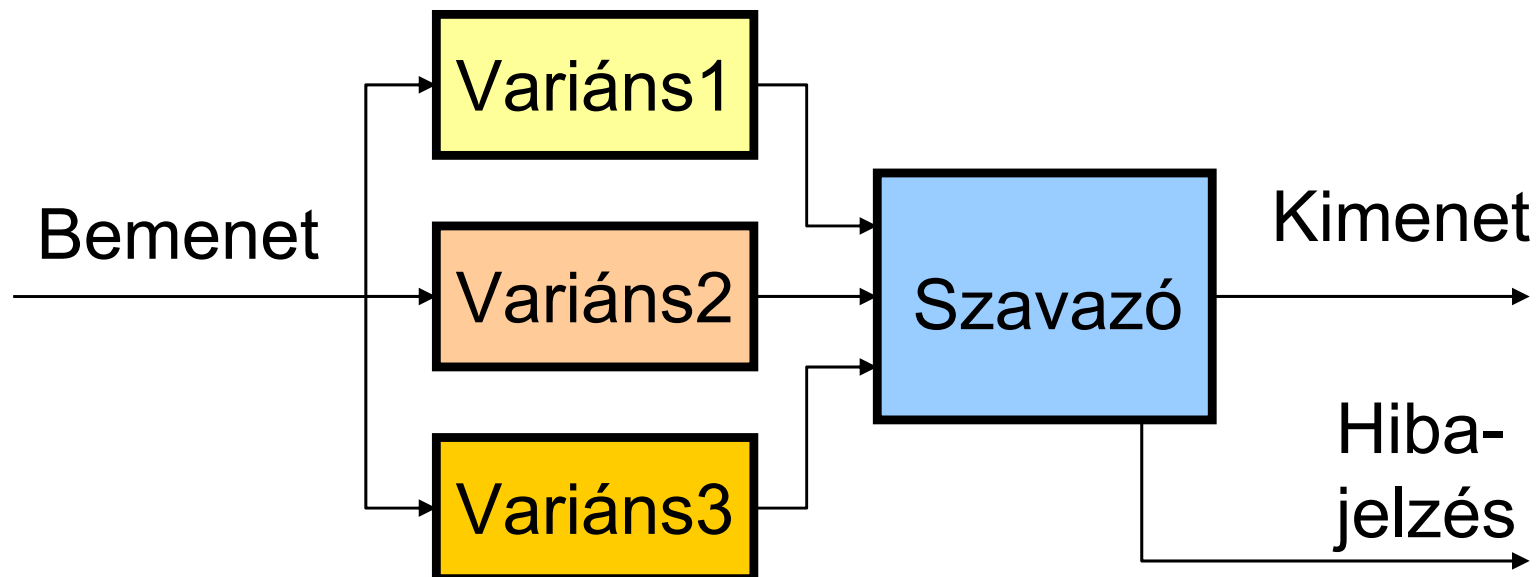
Szoftver tervezési hibák

- Ismételt végrehajtás nem segít (állandósult hiba)
- Redundáns modulok: **Eltérő tervezés** szükséges
- **Variánsok**: azonos specifikáció, de
 - eltérő algoritmus és/vagy adatstruktúrák (diverzitás)
 - más fejlesztési környezet, programnyelv
 - elszigetelt fejlesztés

az azonos hibák bekövetkezésének csökkentésére
- Variánsok végrehajtásának technikái:
 - N-verziós programozás (NVP: N-version programming)
 - Javító blokkok (RB: Recovery block)
 - N-önellenőrző programozás (NSCP: N-self-checking programming)
 - Önkonfiguráló optimista programozás (SCOP: Self-configuring optimistic programming)

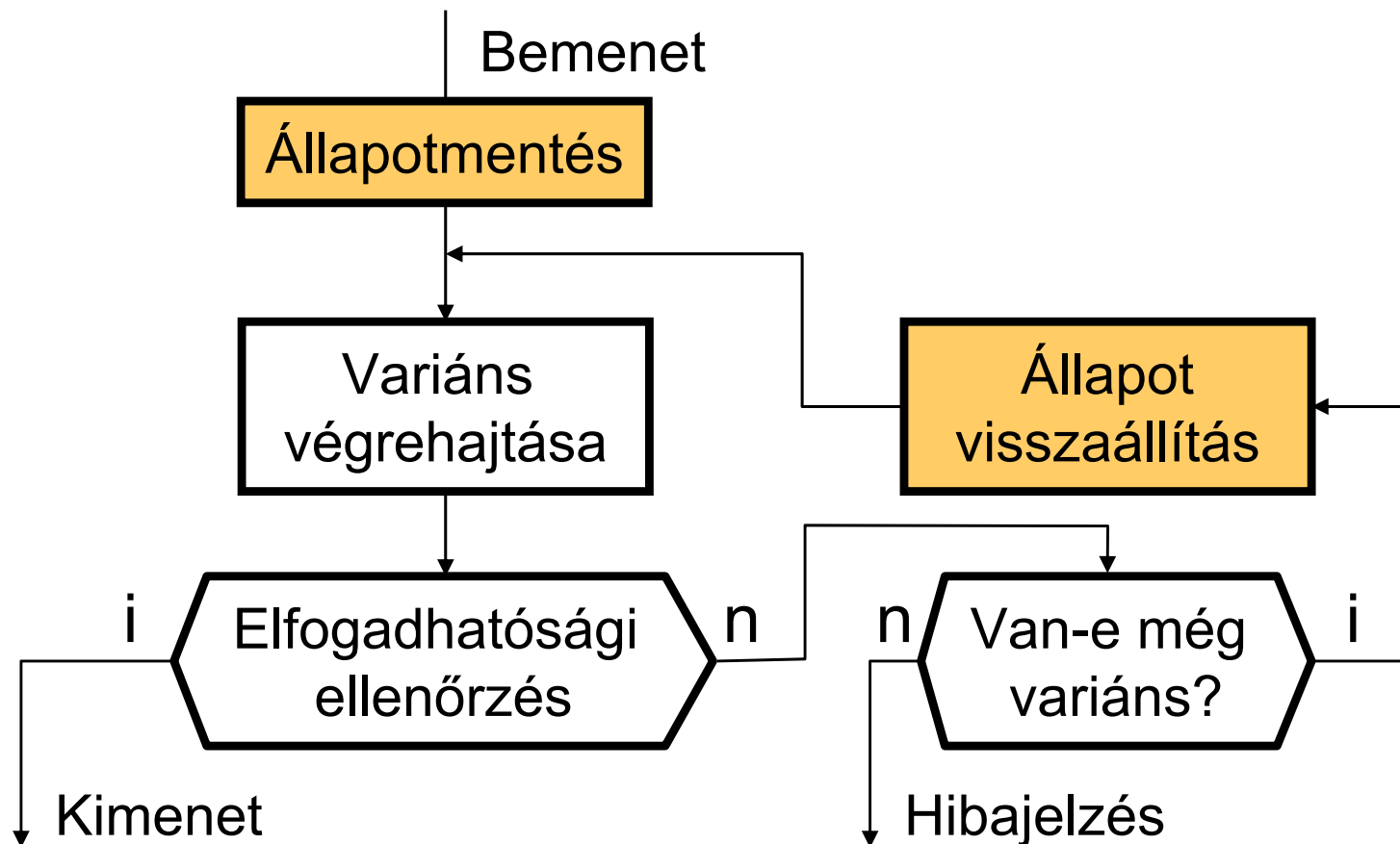
1. N-verziós programozás (NVP)

- **Aktív statikus** redundancia:
Minden variáns végrehatása (párhuzamosan)
 - Ugyanazon bemenetek
 - **Többségi szavazás**
 - Elfogadható értéktartományt kell megadni a kimenetekre
 - Szavazó egyszeres hibapont (SPOF), de egyszerű
- Kritikus komponens: Szavazó és a futtató környezet



2. Javító blokkok (RB)

- **Passzív** redundancia: Csak hiba esetén aktiválódik
 - Variánsok kimenetének **elfogadhatósági ellenőrzése**
 - Hiba esetén tartalék variáns (soros) végrehajtása

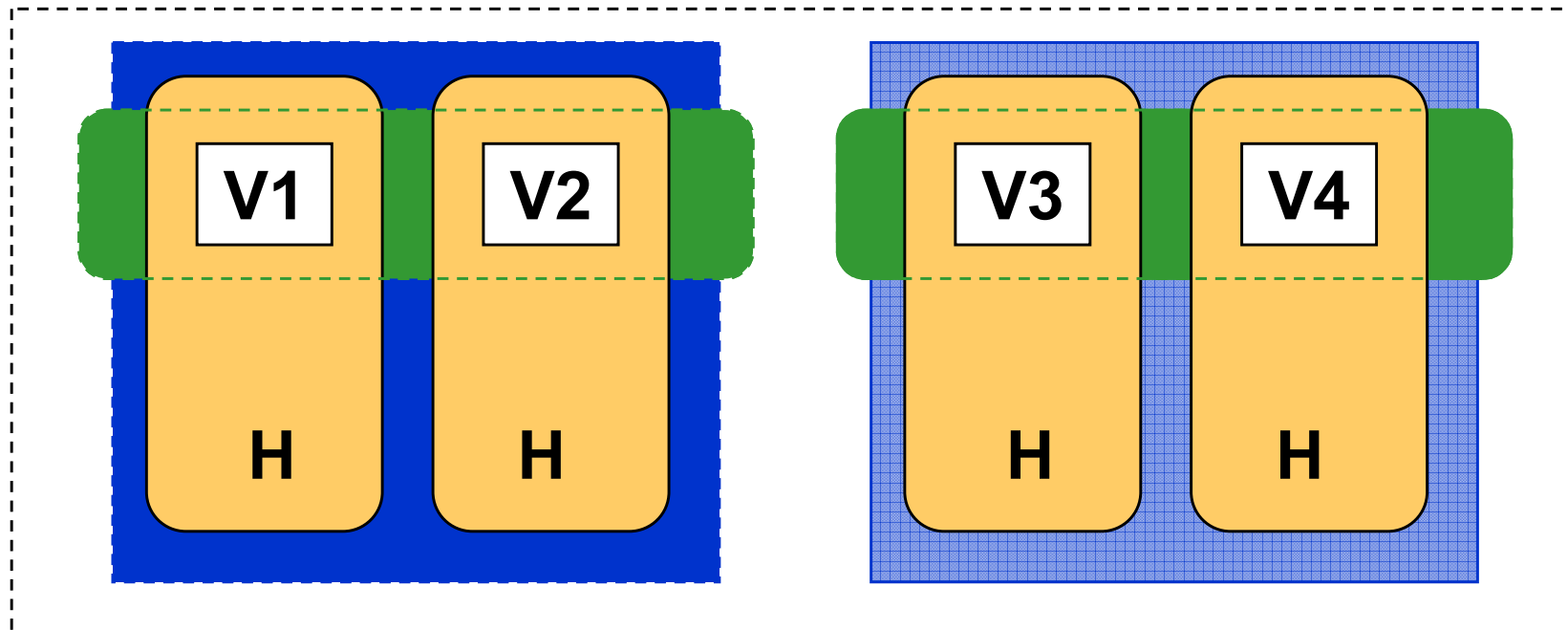


Összehasonlítás

Tulajdonság/típus	N-verziós prg.	Javító blokkok
Ellenőrzés	Szavazás, relatív	Elfogadhatóság abszolút
Végrehajtás	Párhuzamos	Soros
Időigény	Leglassabb variáns (vagy time-out)	Hibák számától függő
Redundancia	Mindig	Csak hiba esetén
Tolerált hibák	$[N/2-1]$	$N-1$
Hibakezelés	Maszkolás	Helyreállítás

3. N-önellenőrző programozás (NSCP)

- **Aktív dinamikus** redundancia:
N számú **önellenőrző** komponens
- Hibadetektálás esetén átkapcsolás az elsődlegesről a (következő) önellenőrző tartalékra
- Jellegzetes példa: Airbus A-320 fedélzeti számítógép
 - Páronként önellenőrző végrehajtás (szoftver variánsok)
 - Elsődleges pár működik, **átkapcsolás hiba esetén**
 - Állandósult hardver hiba: Ismételten hibázó pár **lekapcsol**



4. Önkonfiguráló optimista programozás (SCOP)

Adaptív redundancia: Csak a szükséges számú variáns

- Paraméterek:
 - Hány variáns egyetértése (azonos kimenet) szükséges: **a**
 - Hány variáns van: **n**
 - Mennyi idő áll rendelkezésre: **t**
- Iteratív végrehajtás
 1. Kezdetben a minimális számú (várhatóan egyetértő) variáns végrehajtása: **a** számú variáns
 2. Kimenetek ellenőrzése komparálással: **k** egyezik
 - Ha $k=a$, akkor kilépés: OK
 - Ha $k<a$, akkor (optimista módon) **a-k** újabb variáns végrehajtása az egyetértés eléréséhez, és folytatás a 2. lépéssel
 - Kilépés hibajelzéssel:
Ha nincs több végrehajtható variáns, vagy letelt a **t** idő

Adat diverzitás

- **Alapötlet:**
 - Azonos feldolgozási algoritmus, de **eltérő adatszerkezetű** variánsok (implementáció ezért némileg eltérő)
 - Pl. vektor vagy láncolt lista
 - Pl. más adatbázis kezelő, más lekérdezési mód
 - Másolatok (replikák): Azonos szoftver komponensek, de **eltérő állapotok** lehetnek
 - Pl. numerikus eltérések, más adatbázis példányok
- **Végrehajtási módok:**
 - **Újrapróbáló blokk (retry block):**
 - Javító blokkok analógiája: Passzív redundancia szerinti végrehajtás
 - **N másolat (N-copy programming):**
 - NVP analógiája: Aktív redundancia szerinti (párhuzamos) végrehajtás

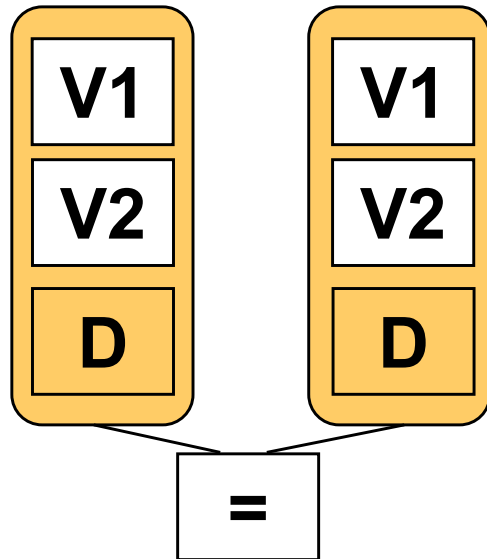
Hibrid architektúrák (áttekintés)

Típus / tolerált **hardver** hibák / tolerált **szoftver** hibák
(maximális) száma

- RB/1/1
 - Elfogadhatósági teszt után **komparálás**
 - Diagnosztikai ellenőrzés után a hibás lekapcsol
- RB/2/1
 - Elfogadhatósági teszt után **szavazás**
 - Ismételten eltérő lekapcsol (RB/1/1 lesz)
- NVP/1/1
 - Variánsok közötti **szavazás**
 - Ismételten eltérő lekapcsol (komparálás marad)
- NVP/2/1
 - 4 variáns esetén **szavazás**
 - Ismételten eltérő lekapcsol, újrakonfigurálás után NVP/1/1 lesz

Hibrid architektúrák 1.

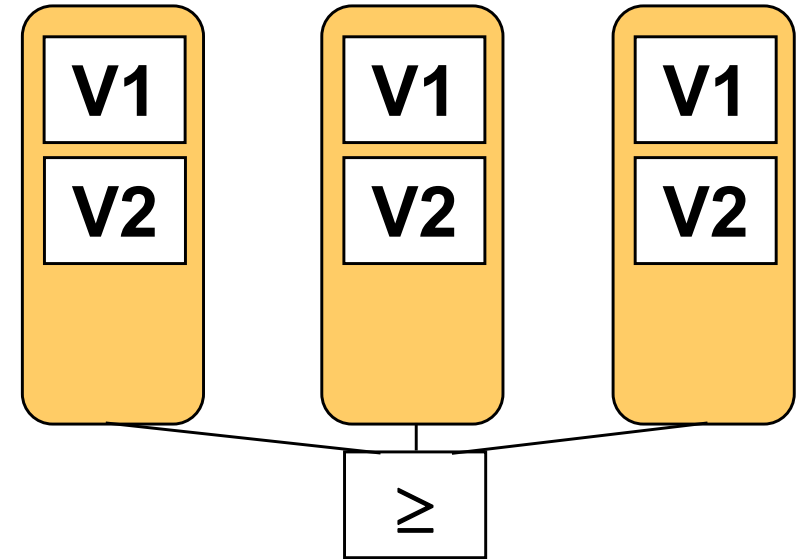
RB/1/1



RB/1/1

- Elfogadhatósági ellenőrzés után **komparálás**
- Diagnosztikai ellenőrzés után a hardver hibás lekapcsol

RB/2/1

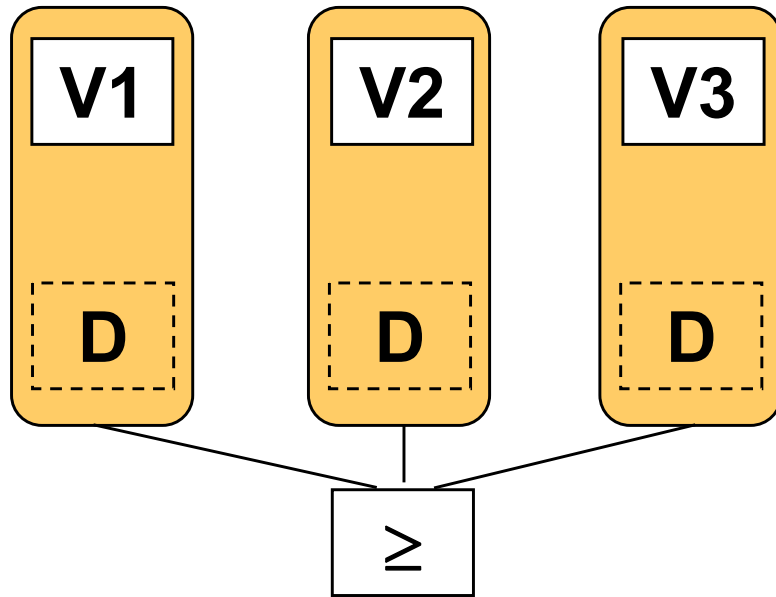


RB/2/1

- Elfogadhatósági ellenőrzés után **szavazás**
- Ismételten eltérő lekapcsol (RB/1/1 marad)

Hibrid architektúrák 2.

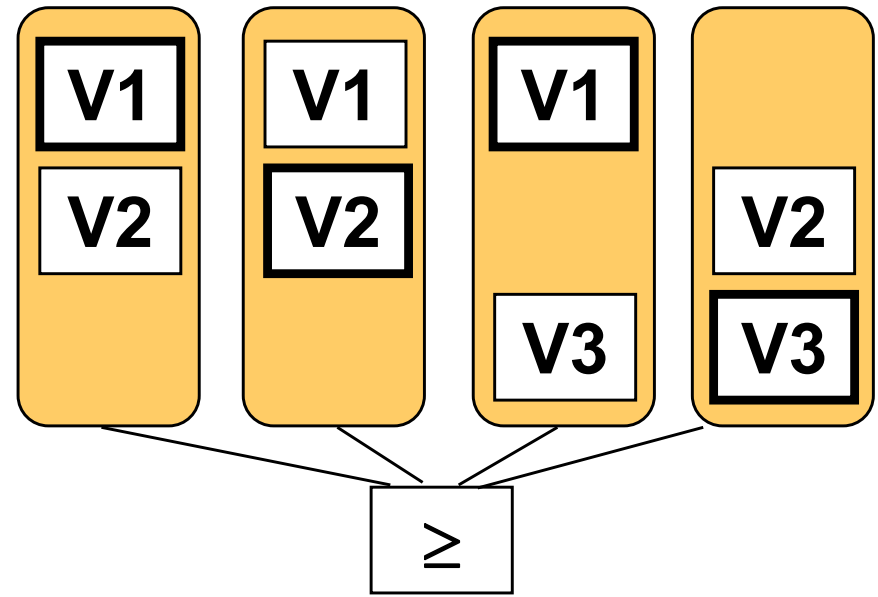
NVP/1/1



NVP/1/1

- Variánsok közötti **szavazás**
- Ismételten eltérő lekapcsol (komparálás marad)

NVP/2/1



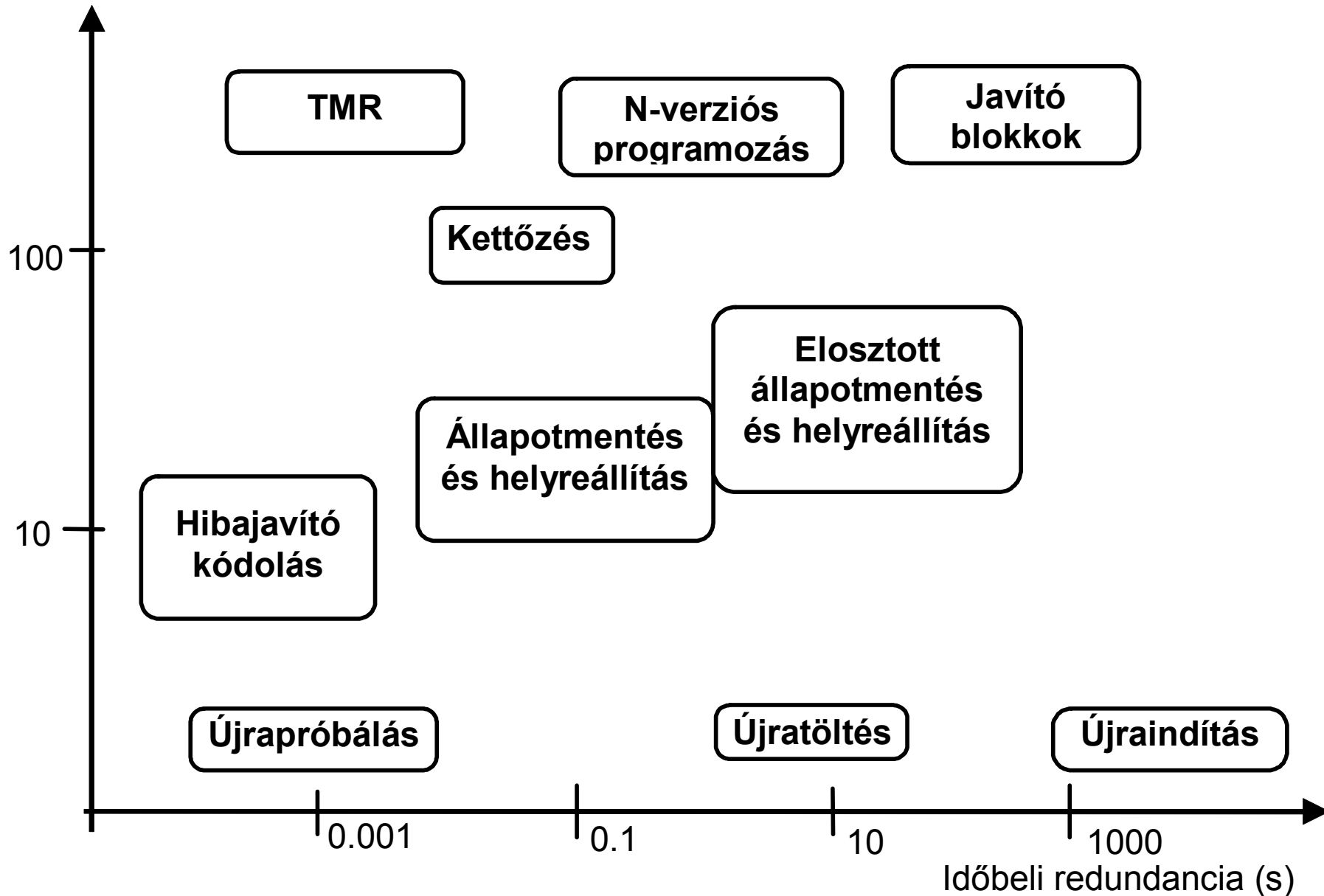
NVP/2/1

- 4 variáns esetén **szavazás**
- Ismételten eltérő lekapcsol (NVP/1/1 konfigurálható)

Összefoglalás

Hibatűrési technikák összehasonlítása

Térbeli redundancia (%)



Szabvány szerinti módszerek

- IEC 61508:
Functional safety in electrical / electronic / programmable electronic safety-related systems
- Szoftver architektúra tervezés

Table A.2 – Software design and development: software architecture design (see 7.4.3)

Technique/Measure*		Ref	SIL1	SIL2	SIL3	SIL4
1	Fault detection and diagnosis	C.3.1	---	R	HR	HR
2	Error detecting and correcting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Safety bag techniques	C.3.4	---	R	R	R
3c	Diverse programming	C.3.5	R	R	R	HR
3d	Recovery block	C.3.6	R	R	R	R
3e	Backward recovery	C.3.7	R	R	R	R
3f	Forward recovery	C.3.8	R	R	R	R
3g	Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h	Memorising executed cases	C.3.10	---	R	R	HR
4	Graceful degradation	C.3.11	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	HR	HR	HR
7b	Semi-formal methods	Table B.7	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8	Computer-aided specification tools	B.2.4	R	R	HR	HR
NOTE – The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in IEC 61508-2.						
* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.						

Összefoglalás: Hibatűrés technikái

1. Állandósult hardver hibák

- Hardver redundancia: többszörözés

3. Időleges hardver hibák

- Szoftver redundancia
 1. Hibadetektálás
 2. Hibahatások felmérése
 3. Helyreállítás: visszalépő vagy előrelépő
 4. Hibaok kezelése
- Információ redundancia: Pl. hibajavító kódolás
- Idő redundancia: Pl. utasítás ismétlés

2. Szoftver tervezési hibák

- Eltérő tervezésű variánsok: Pl. NVP, RB, NSCP, SCOP