MTA Számitástechnikai és Automatizálási Kutató Intézete, **Közlemények** 15/1975.

A SCRATCH–POOL KIHASZNÁLTSÁGÁNAK MÉRÉSE

Tóth Beatrix – Tőke Pál

MTA Számitástechnikai és Automatizálási Kutató Intézete

A CDC-3300-as gépek MASTER operációs rendszere a programok input és output adatait egy speciális disk területén az u.n. scratch pool-on tartja. Ennek a disk területnek a dinamikai kezelése lehetővé teszi, hogy a programok össz igénye – melyet jelenleg előre meg kell adni – meghaladja a fizikai scratch pool területet. Ennek oka, hogy a programok futása során a valódi scratch terület igénye időben véletlenszerüen változik és csak ritkán éri el a maximumot. Több program egyidejü futásánál a véletlen ingadozások miatt az együttes scratch pool igény jóval alatta marad az egyes igények maximumai összegének.

A dolgozatban megvizsgáljuk a scratch igény változását, az egyes jobokra vonatkozóan, mint stochasztikus folyamatot. A statisztikai vizsgálatokon kivül megadjuk a scratch pool jelenlegi kezelésének leirását és megvizsgáljuk a dinamikus kezelés megvalósitásának lehetőségét.

1. A scratch pool kezelési mechanizmusa, vezérlési sajátságai

A scratch pool-n az allokálás minden file esetén szegmens méretben történik. (A szegmens installációs paraméter – jelenleg 1 szegmens = 11 200 szó.) A pool nagysága 250 szegmens.

A *scratch pool belső adminisztrációja* a MASTER operációs rendszerben három rekesz segitségével történik:

PSCRATCHF – a pool méretét tartalmazza szegmensekben

PSCRATCHL – megadja. hogy az adott pillanatban a rendszer mennyi logikai pool igényt képes kielégiteni

OC.TSCNT – megadja, hogy az adott pillanatban a rendszer hány szabad szegmenssel rendelkezik

Egy *job* esetén a teljes logikai pool igény:

$$scr + pun + out + abort$$

ahol mindegyik szegmens méretben értendő. Az igény a job teljes igényét jelenti, illetve a scr igény esetén a fellépő maximális igényt *scr*-vel jelöljük és a többiekre vonatkozóan a t időpontig fellépő igények összegét $pun(t)+out(t)+abort(t)$-vel jelöljük. Látható, hogy ezek időbeli változásának vizsgálata több job együttes futása esetén értékes megtakaritást jelentene, ha minden t időpontban a pillanatnyi job igényt kellene csak lefoglalni a pool számára a disken. Az egyes

igények nagysága nem haladja meg a 63-t (out és abort együtt értendő).
(Ennek oka, hogy a File Definition Table-ban a szegmensszámra 6 bitnyi hely van.) Nyilván a teljes pool igény nem áll fent végig a futás alatt.

A job teljes logikai pool igénye nem haladja meg a PSCRATCHF-t, ellenkező esetben a rendszer a jobot nem schedulálja.

A job inditásánál – mint minden igénynek – a logikai pool igénynek is kielégithetőnek kell lenni, tehát értéke ⩽ PSCRATCHL. A jobok inditására logikailag ez nem szükséges általánosságban ld. pl. [4]. A MASTER által alkalmazott ezen inditási módszer egy lehetséges megoldás a " deadlock " állapotok elkerülésére. Ld. pl. [1]. Az inditásnál történik meg a teljes out, pun, abort igény lefoglalása.

Mivel a scratch pool logikai értékét nagyobbra állitják be a fizikainál, elképzelhető, hogy az allokálandó szegmensszám meghaladja a rendelkezésre álló fizikai szegmensszámot. Ebben az esetben az igényt kibocsátó task speciális SEGTAB WAIT állapotba kerül. Mihelyt az OC.TSCNT tartalma nő, a task ujra kibocsátja az allokálási igényét.

A befejezéskor a TRMOVR, az operációs rendszer egy taskja, a felesleges szegmenseket visszaadja a rendszernek, de az out file -t a nyomtatás, a pun file-t a lyukasztás végéig fenntartja.

A fenti kezelési mechanizmus a következő hiányosságokkal rendelkezik:

Az out file teljes nagyságban már az inditáskor létezik – pedig erre csak később van szükség. A felhasználónak a szükséges sorigény többszörösét adják meg. Végül, a rendszer az igényelt sorokat teljesen kitöltötteknek tekinti, és ennek megfelelően foglalja le a helyet.

Az emlitett problémák megoldására Tőke Pál tett javaslatot [2]-cikkében. Ennek lényege a következő:

a.) kezdetben a rendszer csak egy bizonyos nagyságu out file-t foglalna le a job számára, később ezt szükség szerint növelné.

b.) amennyiben az out file elérne egy előre meghatározott nagyságot, a rendszer ezt automatikusan lezárná és nyomtatná. Ennek az optimális méretét kellene meghatározni. Erre vonatkozó elméleti eredmények találhatók Tomkó [3] dolgozatában. Az optimalizáláshoz szükséges alapvető mérések eredményeit a dolgozatban közöljük.

MASTER 4.0 operációs rendszer már tartalmaz egy PPOF makrot és taskot. Ezt a felhasználónak kell hivnia. Hivás esetén a makro az out file-t két részre osztja: a teleirt szegmenseket lezárja és kinyomtatja, az üres szegmenseket tekinti ezután az out file-nak. Hiányossága ennek a megoldásnak, hogy az inditáskor a teljes file lefoglalás megtörténik, a fölösleges szegmensek a terminálásig megmaradnak.

2. Mérési eredmények

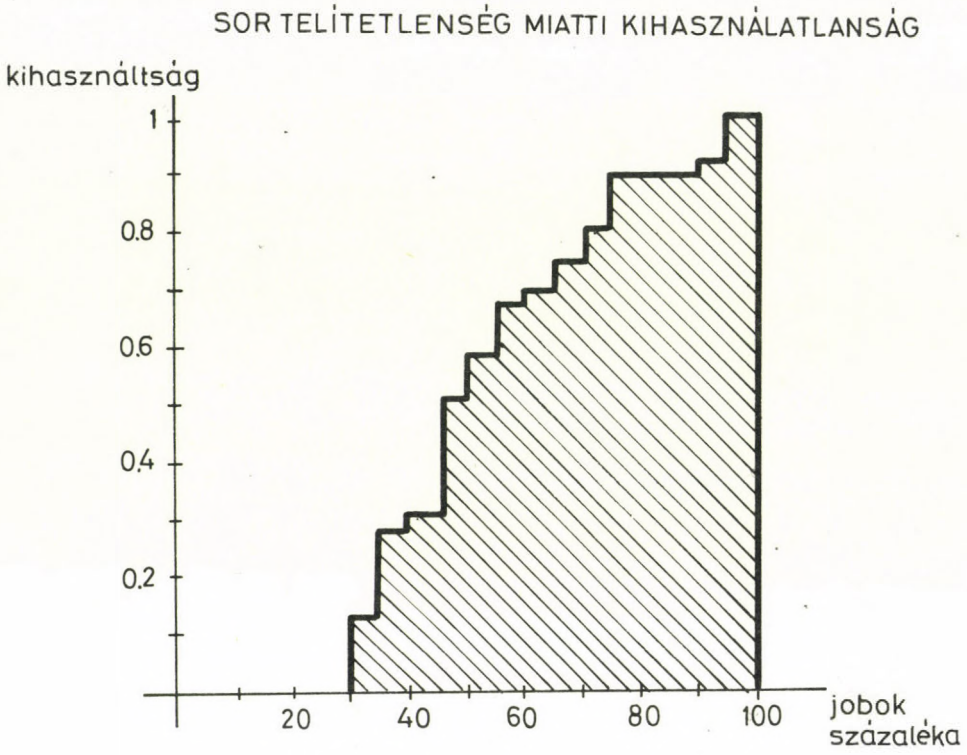
Alapvető célkitüzésünk, hogy az inditáskor az igényeknek megfelelő nagyságu out file-t foglaljunk le, amit szükség esetén bőviteni lehet. A kezdeti szegmensszám meghatározásához mértük a schedulált szegmensek számát, valamint a teleirt blokkok számát. Ugyancsak gyüjti a mérési program a kinyomtatott sorok számát, a kezelési és a terminálási időt, végül a terminálás módját (normál vagy abnormál).

A mérés pontos mechanizmusa. A job terminálásakor a TRMOVR-ben elérhető a job tábla cime. Ennek segitségével végigkerestük a Primary Task List-t (ahol minden program taskhoz és minden nyitott 1/0 file-hoz egy háromszavas bejegyzés tartozik), hogy az illető job file-jának FCI tábla ciméhez hozzájussunk. Majd ebből a megfelelő File Definition Table cimét olvastuk ki. Ez utóbbiból kaptuk adatainkat.

A mérés megvalósitása. A méréseket 1974. októberében normál user futások alatt végeztük kb. 21 óra hosszat (5 alkalommal). Ez idő alatt a lefutott jobok száma 646 volt.
Méréseink két hiányosságát emlitem meg.

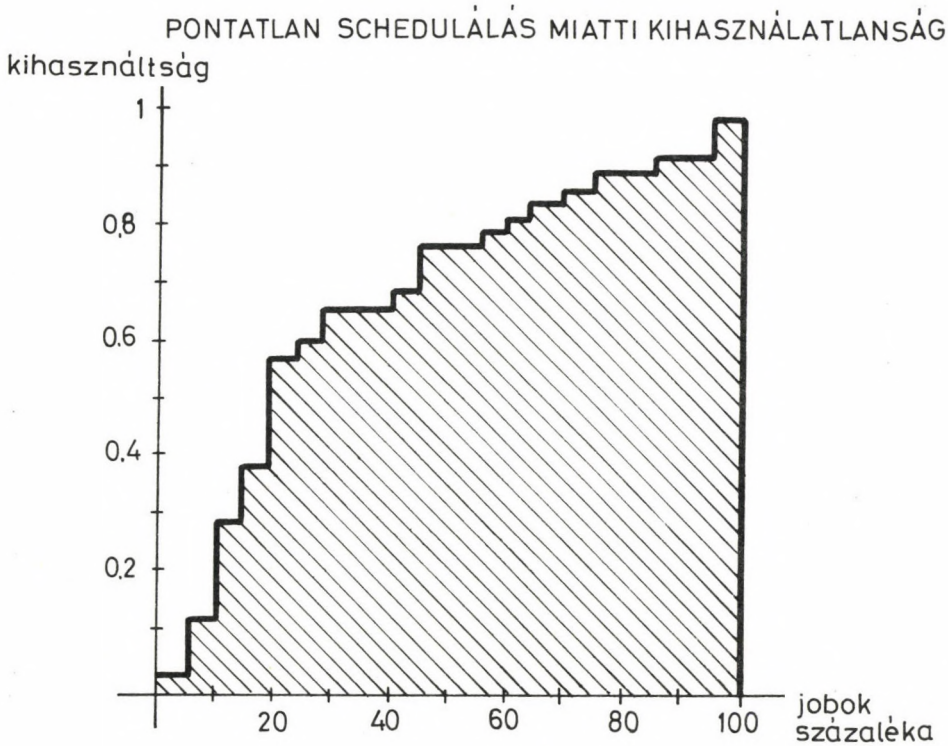
Ugy tekintettük, hogy az out file a terminálásig létezik, pedig a printelés befejezéséig megmarad – de ez utóbbi időpontot nem tudjuk pontosan meghatározni.
Az abnormálisan terminált jobokat figyelmen kivül hagytuk. Ennek oka az volt, hogy az abort paraméter használata lényegesen megváltoztatja az out file kihasználtságot.

Vizsgáltuk a *sorok telitettségét.* Ezen mutató meghatározásánál a következő mérési adatok álltak rendelkezésünkre: Ismerjük a kinyomtatott sorok számát, továbbá azon disk - terület nagyságát, amelyen ezen sorok ténylegesen tárolódnak. Az operációs rendszer maximálisan teleírt sorokat tételez fel, amikor kijelöli az OUT file méretét. E két adat, a ténylegesen mért block-szám továbbá az adott sormennyiségnek megfelelő disk-terület egymáshoz való viszonyitása adja a sortelítetlenségi mutatót. Ennek várható értékére 0,53-at szórására pedig σ^2_T = 0,035-öt kaptunk. Az empirikus eloszlásfüggvényt az alábbiakban szemléltetjük.

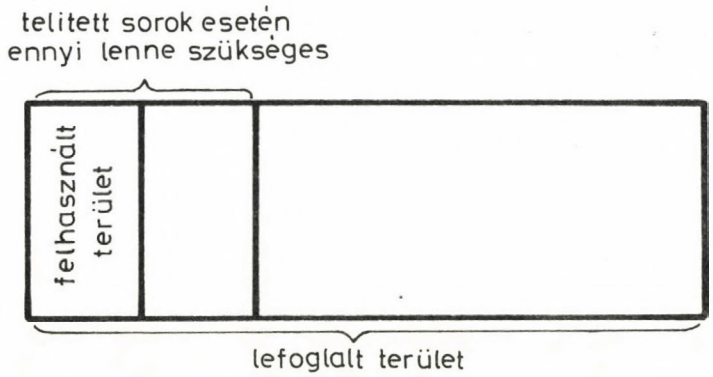


1.ábra

A pontatlan schedulálás mutatóját a kinyomtatott sorokhoz szükséges szegmensek számának, és a ténylegesen lefoglalt szegmensszámnak a hányadosából kapjuk. A sorok számából a szegmensméretre való áttérésnél az operációs rendszer schedulálási algoritmusát vettük alapul, így a sorokat teljesen kitöltöttnek tételeztük fel. Ez a mutató átlagban 0,35 volt, tehát durván háromszor nagyobb terület foglalódik le a diskben indulásnál még ideális esetben is (teljes sortelítettség feltételezése) mint amennyi valójában szükséges. Ha még figyelembe vesszük a sortelitetlenségi defektust is, akkor ez az ellentmondásos helyzet még élesebben rajzolódik ki. Ezen kihasználtsági mutató szórása $\sigma^2_T = 0{,}07$-nek adódik. Az alábbi ábra empirikus eloszlásfüggvényt szemlélteti.

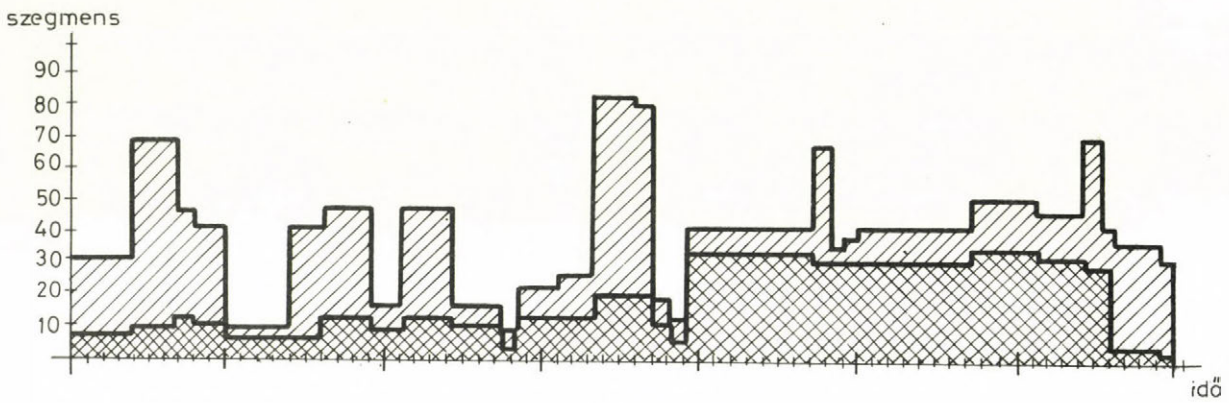

PONTATLAN SCHEDULÁLÁS MIATTI KIHASZNÁLATLANSÁG
kihasználtság
1
0.8
0.6
0.4
0.2
20
40
60
80
100
jobok
százaléka

2.ábra.


telített sorok esetén
ennyi lenne szükséges
felhasznált
terület
lefoglalt terület

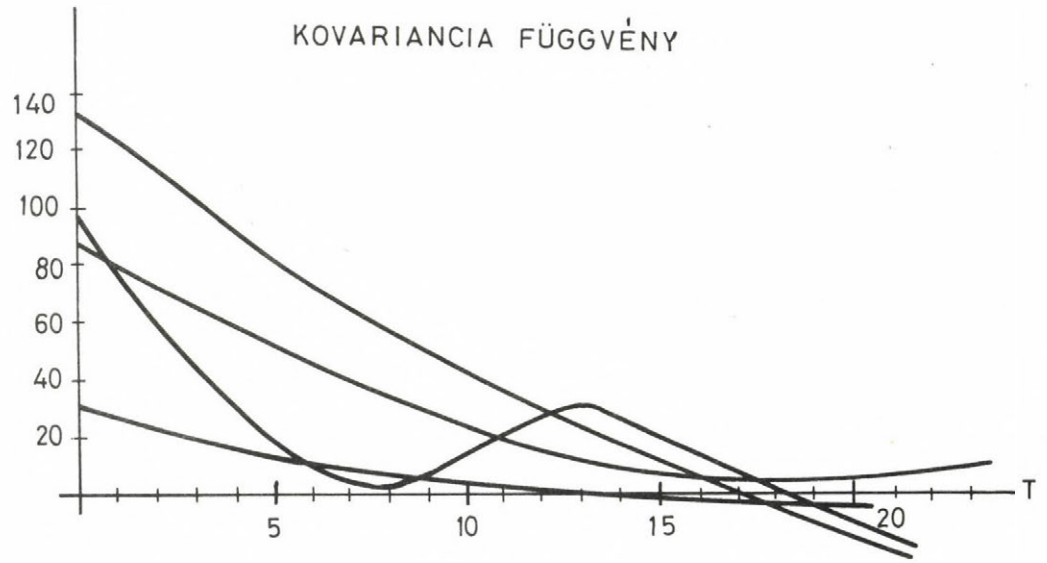
3.ábra.

A következő ábra az out igények alakulását mutatja: a lefoglaltat és a ténylegest. A mintát percenként vettük kb. 5 óra hosszat (– itt 70 percet ábrázoltunk). Hosszabb mérés esetén ez a folyamat sztochasztikus változást mutat és időben folytonosnak vehető. Első közelitésben egy 2 dim-s Gauss–Markov folyamatnak tekinthetjük. Ha a későbbiekben ez jelentős eltérést mutat, akkor korrigáljuk.



4. ábra.

Az empirikus kovarancia függvény azt mutatja, hogy 16-18 percenként gyakorlatilag már nincs korreláció:



5 ábra.

A helyfoglalás, mint emlitettük, szegmensenként történik. Egyszerű mutatóval próbáltuk objektive mérni, hogy a szegmensméret megválasztása (installálási probléma) mennyire durva. Lényegileg a "túlschedulálást", ami a lefoglalás szegmensnyi kvantumos voltából adódott, mértük *sor-egységben* és viszonyitottuk a tényleges sor-igényhez. Nyilván ez a viszonyszám kisebb scr-igényü joboknál nagyobb, mig a nagy nyomtatási igény mellett elenyésző.
A szegmensméret megválasztásának jóságát éppen az adja, hogy ez a mutató mikor válik megnyugtatóan kicsivé. Gyakorlatilag 8 lefoglalt szegmensnél ez a mutató 0,0025% körül van, ami elfogadható. (8 szegmens kb. 2000 sor-nak felel meg).

A fenti mérési eredmények elegendő alapot szolgáltatnak arra, hogy sokkal dinamikusabb SCRATCH–POOL kezelő mechanizmust dolgozzunk ki. A kapott eredmények szimulációs módszerek alapjául szolgálhatnak, mivel a valóságos helyzetet tükrözik.

Elképzeléseink az új kezelési politika gyakorlati megvalósitására már készek. Jelenleg a MASTER 4.0 operációs rendszerhez történő illesztésén fáradozunk.

Irodalom

[1] Colin, A.S.T.; Introduction to operating systems, 14. fejezet, Mac Donald (American Elsevier) London, New York, (1971).

[2] Tőke P., Dynamic use of the scratch-pool, Proceedings of the Computer Science Conference, 117-121 pp. Székesfehérvár (1973).

[3] Tomkó J., Studynig Output Organization with a Single Finite Quence, Proc of. Conputer Science Conference, 113 - 117 pp. Székesfehérvár (1973).

[4] IBM. OS/360 Consepts and Facilites. In Programming Lauguages and Systems. (S. Rosen, Ed.) Mc Graw - Hill, 1967, 598 pp. New York.

Summary

A scratch – pool utilization study

B. Tóth, P. Tőke

The MASTER operating system, for the CDC 3300, keeps the job's input, output and scratch files on a common disk are called SCRATCH-POOL. In our paper we discuss the management of this pool and the statistical investigation about the exploitation of the OUT file (fullness of lines, defekt in consequence of the inaccurate scheduling, examination of the segment size).

On the basis of our results we think possible the elaboration of a more dynamic SCRATCH-POOL management.

Р е з ю м е

Об эффективности дисковой памяти операционной системы MASTER

Б. Тот, П. Токе

Операционная система MASTER для машины CDC-3300 распределяет дисковую память из общего массива ОС так называемого SCRATCH-POOL для вводного массива с перфокарт, для выводного на печать, и для рабочих массивов заданий.

В стате затрагиваются вопросы методического обращения и управления общего массива ОС, а так же произвадится статистический анализ использования выводного массива на печать (уплотнение строк, дефекты из-за неточного задания числа строк, выбор размера сегмента и т.и.).

На основе разработанных данных становится возможным осуществление более динамического механизма управления.