MTA Számitástechnikai és Automatizálási Kutató Intézete, Közlemények 18/1977,

DINAMIKUS ERŐFORRÁS ELOSZTÁS VEGYES REAL–TIME ÉS BATCH–ÜZEM ESETÉN

Gyarmati Péter
KSH Számitástechnikai Igazgatóság

1. Bevezetés

A számitástechnika mai állása lehetőséget nyujt arra, hogy a különböző feladatokat a nekik megfelelő legalkalmasabb üzemmódban hajthassuk végre. A különböző üzemmódokat legcélszerübben külön-külön számitógépeken lehetne megvalósitani, azonban erre kereskedelmi, illetve gazdasági okok miatt általában nincs mód. Mivel korunk számitógépének kapacitása, erőforráskészlete, sebessége igen nagy, meg van a lehetőség arra, hogy a különböző üzemmódokat egyazon gépen valósitsuk meg.

A távfeldolgozási igény egyre jobban terjed, de még ma is jelentős mennyiségü a hagyományos módon végzett feldolgozás. Ennek a kettős igénynek a kombinációja adja a vegyes, real-time és batch üzem felállitásának szükségességét. Feltéve azt, hogy már rendelkezünk a vegyes üzemmódhoz szükséges, alapvető hardware és software eszközökkel, akkor még mindig biztositanunk kell a beérkező feladatok párhuzamos, egyidejü és ami a legfontosabb – igény szerinti futtatását.

Ennek a problémának a következő – egymásnak potenciálisan ellentmondó – teljesitmény tényezői vannak:

- a real–time üzemhez szükséges válaszadási idő (response time);
- a batch üzemü feladatok átfutási ideje (turnaround time);
- a rendelkezésre álló erőforrások minél jobb kihasználása a költségek csökkentése érdekében (throughput).

Az általában rendelkezésre álló ütemezési algoritmusok egy-egy tényezőt kielégithetnek, de többnyire ellentmondanak a többi tényezőnek.

Például egy maximális erőforrás kihasználásra törekvő algoritmus teljesen megváltoztathatja, tetszőlegesen hosszura növelheti a real-time üzem válaszadási idejét.

Mivel a rendszer tervezésekor általában nem ismerjük teljes pontossággal az igényeket, az üzemet csak valamilyen valószinüséggel tervezhetjük meg, amelyhez képest a ténylegesen állapotok elérhetnek.
Különösen vonatkozik ez a csúcs terhelésü időszakokra.

Célszerü tehát egy rugalmasan változtatható, változó, dinamikus erőforrás elosztási algoritmus kialakitása, amely a valószinüségi alapon tervezett rendszert igyekszik a pillanatnyi igények szerinti optimális módon szabályozni.

Az előadás célja egy ilyen dinamikus mérő és szabályozó algoritmus tapasztalatokon alapuló, logikus megközelitése.

2. A cél kitüzése

Először rögzitsük a batch, illetve a real-time feldolgozás rövid definicióját. Ez a definició nem lesz teljesen pontos, de elegendő az előadásban felmerülő kérdések megértéséhez, tárgyalásához. A definició készithető akár az idő, akár a feldolgozandó adat alapján.

Definició a feldolgozandó adat alapján

Ha a feldolgozási igény az adat keletkezésével egyszerre lép fel, akkor *real-time* feldolgozásról beszélünk.
Ha az adatokat elöbb adatállományba összegyüjtjük, majd egyszerre dolgozzuk fel, akkor *batch* feldolgozásról beszélünk.

Definició az időre alapozva

Ha a feldolgozás ugyanabban az időben és időléptékben megy végbe, mint a feldolgozással modellezett valóságé, akkor *real-time* feldolgozásról beszélünk.
Ha a feldolgozás ideje és időtartalma a modellezett valóságtól független, akkor *batch* feldolgozásról beszélünk.

A két feldolgozási mód alkalmazása egyazon gépen nem uj probléma, de a mai fejlettségi állapot egy ujabb megközelitést tesz lehetővé.

A vegyes üzemre történő átálláskor a legalapvetőbb alrendszereket át kell értékelni, esetleg teljesen ujra kell tervezni.
Ezek:

- az adatok tárolásának és hozzáférésének rendszere;
- az adatgyüjtés és adatbevitel;
- a feldolgozási igények kezelése és irányitás;
- a számitógép kezelése és üzemeltetése.

Témánk az utóbbi két alrendszerrel kapcsolatos, pontosabban fogalmazva egy olyan *algoritmus felállitása a számitógép üzemeltetésére, amely a feldolgozási igények kezeléséből fakadó szempontokat a lehető legjobban elégiti ki.*
A cél tehát, hogy elfogadható határok között tartsuk

- a beatch jobok átfutási idejét;
- a real-time utasitások válaszadási idejét;
- a számitógép legkedvezőbb kihasználását;

az erőforrások elosztásának *igény szerinti* irányitásával. A megvalósitáshoz abból az állitásból kell kiindulnunk, hogy a felhasználói igények és a számitógép üzeme egymástól független, annak

ellenére, hogy a gép konfigurációját és az üzemi paramétereit egy adott időszak felhasználói igénye alapján igyekszünk megvalósitani.

Különböző kompromisszumokat, az időt és a felhasználói igények változását figyelembevéve mindig kiderül, hogy a tervezett rendszer végül fázis késésben van. Ha a rendszerünket sikerül elegendően adaptivvá tennünk az emlitett változásokra, akkor a fázis késés – a kapacitás problémáján kivül – megszüntethető. Az adaptiv vezérlés egyik módja az erőforrások felhasználói igény szerinti minél dinamikusabb elosztása. Elvileg minden idő pillanatban az erőforrások ugy oszlanak meg, hogy mindig a legjobban "rászorult" igények jussanak hozzá.

Ehhez két fontos, alapvető tényező szükséges:

– az erőforrások megfelelő eloszthatósága;

– a felhasználói igények pilllanatnyi "rászorultságá-"nak ismerete.

A két tényező ismeretében az algoritmus már nagyon egyszerü:

1.) Megállapitandó a felhasználói igények pillanatnyi "rászorultsága".

2.) Az erőforrások elosztása a "rászorultság" rendjében.

3.) Ha elértük a kapacitás határát a "legkevésbé rászorultakat" késleltetjük és visszatérünk az 1. pontra.

3. Az erőforrások eloszthatósága

A dinamikus algoritmus megvalósitásához a gép erőforrásainak dinamikusan ujra eloszthatónak kell lenniök, tehát olyan számitógépre és operációs rendszerre van szükség, amely ezt lehetővé teszi.

Ennek vizsgálata a jelen előadásnak nem célja, tehát feltételezzük, hogy ez teljesül a legfontosabb erőforrások – CPU idő, input-output utasitások, központi tár – quantitativ mérési lehetőségével együtt.

4. A felhasználói igények pillanatnyi "rászorultsága"

A rendszer tervezési szakaszában a felhasználói igények ismeretében különböző *teljesitmény-csoportokat* állitunk fel, amelyeket erőforrás felhasználás szempontjából különböző küszöbértékekkel és paraméterekkel jellemezhetünk. Ha futtatás közben a felhasznált erőforrásokat hasonló értékekben mérjük, akkor ezek összevetésével megállapitott eltérés egy olyan mennyiséget ad, amely jellemző az igény pillanatnyi erőforrás "rászorultságára" és igy a teljesitmény-csoportjának megfelelő kiszolgálásban részesülhet.

A mérés és a vizsgálat elvégzéséhez az alábbi definiciókban leirtak szükségesek.

4.1. **Definició.** Kiszolgálási mérték (S)
Egy adott időintervallumban a felhasználói igény által használt erőforrások sulyozott mennyisége időegységre vonatkoztatva,

$$S = A(CPU) + B(MEM) + C(i/0)$$

– A, B, C rendszerparaméterek az erőforrások sulyozására.

– Az erőforrások mennyiségi mérése sokféleképpen oldható meg.

Egy példa:

$$(CPU) = \frac{1}{t} \cdot \frac{\sum_t t_{cpu}}{t_{const}}$$

ahol $\sum_t t_{CPU}$ a t időintervallumban felhasznált *CPU* idő és t_{const} az egységnyi *CPU* idő, pl: 1000 átlagos *CPU* utasitás végrehajtásához szükséges idő.

$$(MEM) = (CPU) \cdot n_{page}\Big/_t$$

ahol n_{page} a t időintervallum végén aktiv központi tároló lapok száma.

$$(i/0) = \frac{1}{t} \sum_t n_{command}$$

a t időintervallumban végrehajtott $i/0$ utasitások száma.

4.2. **Definició.** Mérési időintervallum (t)

A dinamikus algoritmus egyes végrehajtásai között eltelt idő. Az egyes időintervallumok különböző hosszuságuak lehetnek, kezdetét, vagyis az algoritmus végrehajtását az alábbi események határozzák meg:

– CPU várakozó állapotba kerül:

– page fault (központi memória lapváltás igénye):

– uj igény belépésekor.

4.3. **Definició.** Teljesitmény mérték (P)

Minden teljesitmény-csoporthoz tartozik egy teljesitmény mérték, amely szerint az ide belépő felhasználói igény futtatásakor erőforrást kér. A kiszolgálási mértékkel azonos léptékü és dimenzióju.

4.4. **Tétel.**

1. $P < S$ *a kiszolgálás magasabb szintü az igenyeltnél, KÉSLELTETHETŐ!*
2. $P = s$ *a kiszolgálás megfelel az igénynek*
3. $P > S$ *a kiszolgálás alacsonyabb szintü az igényeltnél, KÉSIK!*

5. **Algoritmus.**

Minden feldolgozási igény a három döntés alapján kerül besorolásra a következő időintervallum idejére az alábbi módon:

1. Az igény a *várakozó sorba kerül* (SWAP OUT) jelzést kapja az (S – P) érték nagyságának növekvő sorrendjében.
2. Az igény a *kiszolgálás folytatódik* jelzést kapja, ha a várakozó sorban volt, akkor az *aktiválásra kerül* jelzést kapja (SWAP IN) a várakozó sorban elfoglalt helyének sorrendjében.
3. Az igény a *kiszolgálás folytatódik a 2. szerinti igényeket megelőzve* jelzést kapja a (P – S) érték nagyságának csökkenő sorrendjében; ha a várakozó sorban volt, akkor az *aktiválásra* (SWAP IN) *kerül* jelzést kapja a (P – S) érték nagyságának megfelelő besorolással.

Az ezután következő *SWAP algoritmus* fogja az igényeket a központi tár, illetve a virtuális tár lehetőségeinek megfelelően elhelyezni.

A SWAP algoritmus a tárkapacitástól függően a kiszolgálás folytatódik jelzésüeket is a várakozó sorba helyezhet, illetve a várakozó sorba kerül jelzésüeket is aktivan hagyhat.

Az algoritmus végsősoron a felhasználói igényeket minden egyes időintervallum végén a kiszolgálástól függően – az (S – P) érték nagysága szerint – sorba rendezi, majd a SWAP algoritmus a tár kapacitásának megfelelő számu – a sorban elől álló – igényt aktivál.

Néhány megjegyzés az algoritmushoz

Nem feltétlenül jelent túlterhelést, ha vannak igények a várakozó sorban (SWAP OUT)

A pillanatnyi túlterhelés nem biztos, hogy késést okoz a külső ismérvek (response, turnaorund time) alapján.

A rövid idejü túlterhelések kiegyenlitődnek, mert mindig azok kerülnek késleltetésre, amelyek a legjobban elviselik.

A rendszer hatékonysága akkor érvényesül, ha a belépő összes igény azonnal aktiválásra kerül. Ennek csak a SWAP algoritmus gazdaságossága szab határt.

Az algoritmus lehetőséget ad az optimális gépkihasználásra, mivel csak az időintervallum határokon lép be és csak a feldolgozási igények végrehajtási sorrendjét változtatja meg.

A megoldás a SWAP algoritmus szempontjából nem optimális, azaz lehetséges olyan helyzet, amikor feldolgozási igények sokszor mozognak a központi tár és a várakozó sor között (SWAP–IN, SWAP–OUT).

A késleltetések, illetve a késések elosztása az egyes igények között nem befolyásolható, mert ezt az algoritmus a pillanatnyi állapotok szerint intézi. Ez huzamos idejü tulterhelés esetén okozhat gondot.

Az emlitett problémák megoldása az algoritmus továbbfejlesztésével lehetséges.

6. Az algoritmus továbbfejlesztése

A számitógép rendszer sokszor olyan túlterhelési állapotba kerül, amikor a késleltetések már a külső ismérvek szerinti rendszeres késésekben jelentkeznek. Ilyenkor külső beavatkozás szükséges, amellyel a késések, vagy a késleltetések elosztását szabályozni lehet.

A szabályozás bevezetésével meg lehet határozni különböző terhelési állapotok esetére a késleltetések elosztását az egyes teljesitmény csoportok között.

A megoldáshoz be kell vezetni a *terhelési szint* (W) fogalmát, valamint ujabb paraméterekkel kell kiegésziteni a *teljesitmény-csoport* leirását.

6.1. **Definició.** Terhelési szint (W)
Az a mérték, amely megadja, hogy egy adott időintervallumban az összes kiszolgálási igény hányszorosa a felhasznált kiszolgálásnak.

$$W = \frac{\sum_i P_i(t)}{\sum_i S_i(t)}$$

ahol i az aktiv igények száma.

6.2. **Definició.** Teljesitmény csoport (Q)
Azon ismérvek összesége, amelyek meghatározzák, hogy egy felhasználói igény milyen módon juthat végrehajtása során erőforrásokhoz.

Az alap algoritmus esetén a teljesitmény-csoport $Q(t) = f(P,S(t))$ függvénnyel irható le.

A továbbfejlesztésben a függvény további változókkal bövül:

$$Q(t) = f(P,S(t), W(t),T)$$

ahol P a teljesitmény mérték, S a kiszolgálási mérték, W a terhelési szint, T a külső szabályozás szerinti periódus idő.

A Q függvény változóihoz különböző küszöbértékeket rendelve irhatjuk le az egyes teljesitmény-csoportokat.
Az i-edik csoport $Q_i(t) = f(P_i,S_i(t),W(t),T_j)$

Az információs függvény azt fejezi ki, hogy a P_i kiszolgálási igénnyel belépő feladat a T_j periódusban, $W(t)$ gépterhelési körülmények között, $S_i(t)$ kiszolgálásban részesült a t időintervallumban.

A terhelési szinttől való függés leirása

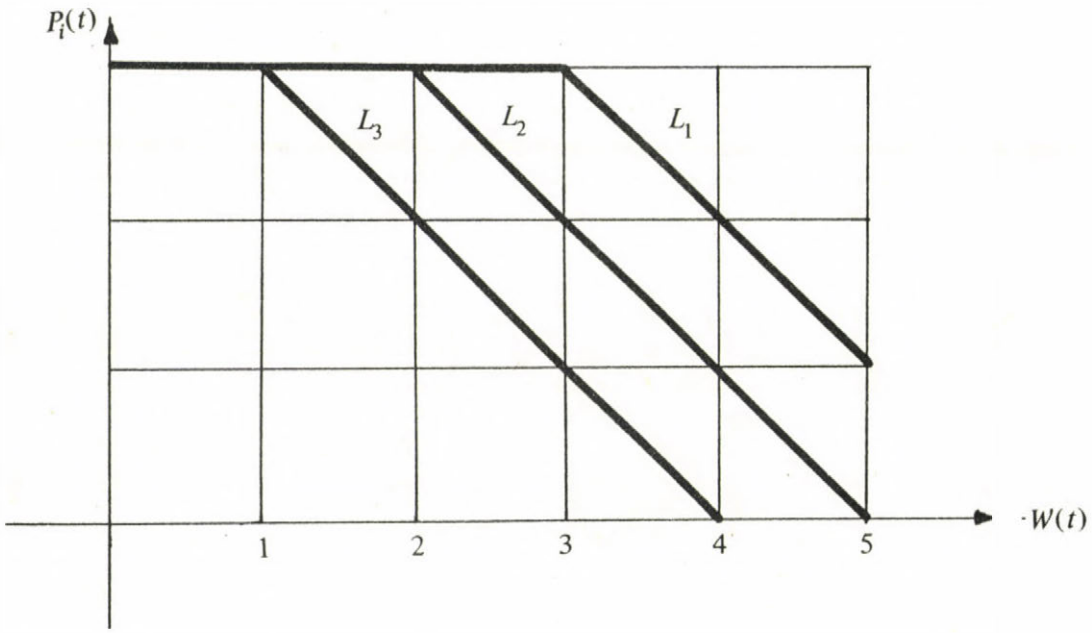
Eredeti célunk szerint azt kivánjuk elérni, hogy a különböző terhelések esetén bekövetkező késleltetéseket előre tervezett módon osszuk el az egyes teljesitmény-csoportok között.

A feladatot ugy oldhatjuk meg legegyszerübben, hogy a P_i értéket tesszük függővé a terheléstől, illetve a külső szabályozás periodusától:

$$P_i(t) = f(W(t),T_j)$$

ahol a $P_i(t_0) = P_i$, vagyis a feldolgozási igény iniciális teljesitmény mértéke.

A függvény azt fejezi ki, hogy az i-edik teljesitmény csoportban lévő feldolgozási igények a t időintervallumban $P_i(t)$ kiszolgálást igényelnek és ennek értéke a pillanatnyi terhelési szinttől és a külső szabályozási periódustól függ, kivéve a feldolgozás első, belépő esetét,amikor a függvény a P_i kezdeti, maximális értéket veszi fel.



Tehát $P_i(t)$ értékét a terhelési függvényében ilyen görbével lehet leirni. Az L segédparaméter felvételével egy görbesereget irhatunk le és ez lehetővé teszi a T_j periódussal való összekapcsolást az alábbi módon:

a Q_i teljesitmény-csoport $P_i(t)$ kiszolgálási igénye

a T_j periódusban, a $W(t)$ függvényében L_k szerint alakul.

Például:

Q_1 teljesitmény-csoport:

- p_1 kezdeti kiszolgálási igény;
- T_1 = 600 sec periódusban
- kiszolgálás L_2 szerint;
- T_2 = 60 sec periódusban
- kiszolgálás L_1 szerint;
- T_3 = futtatás befejezéséig periódusban
- kiszolgálás L_3 szerint

Példa a teljesitmény-csoportok leirására a felhasználók felé:

Q_1 = átlagos batch job-ok (2 óra átfutási idő, ha az adatok mennyisége nem haladja meg a 20 ezer tételt és a gép terhelése közepes).

Q_2 = egyéb batch job-ok (24-48 óra átfutási idővel).

Q_3 = real-time commandok (lekérdező terminálokról, 30 másodperc válaszadási idővel reggel 8-tól 11-ig).

Q_4 = real-time commandok és gyors batch jobok (CRJE terminálokról 2 perc válaszadási idővel).

Q_5 = real-time commandok (interaktiv programozási helyekről a preferált programnyelvekben 2 másodperc válaszadási idővel).

7. Összefoglalás

Az algoritmus az előre meghatározott igénnyel belépő feldolgozási feladatokat igyekszik a számitógép terhelési viszonyainak megfelelően a lehető leggyorsabban végrehajtani, amelyet a felhasznált erőforrások mennyiségének vizsgálatával ér el.

Lehetőséget ad az A, B, C rendszer paraméterek és a P_i küszöbértékek alkalmas megválasztásával az adott konfigurációhoz és feldolgozási igényekhez való illesztésre.

Biztositja a túlterheléssel müködő számitógépen bekövetkező késések megfelelő elosztásának lehetőségét a T_j és L_k paraméterek segitségével.

A müködéshez szükséges változók és paraméterek értékeinek megfelelő rögzitésével mérési adatokat bocsát rendelkezésre a rendszer üzemének vizsgálatához.

A vizsgálatok eredménye alapján az algoritmus egyszerüen módositható a már emlitett paraméterek és küszöbértékek változtatásával.

Az algoritmus gépidő terhelése az aszinkron alkalmazással minimális (a CPU várakozó állapotba kerülésekor fut, vagy amikor amugyis valamilyen ütemezési algoritmus futna). Ez egyben azt is jelenti, hogy a t időintervallum hosszusága nem tartható kézben, de erre valószinüleg nincs is szükség (bizonyitandó).

Külön problémát jelent az algoritmus beépitése egy adott számitógép és operációs rendszer környezetbe.

Az első modellezési eredmények azt mutatják, hogy az algoritmus kedvezően alkalmazható olyan számitógép környezetben ahol virtuális memória (gyors SWAP), többszörös, vagy elegendően gyors $i/0$ csatornák és a külső periféria (printer, stb.) vezérlésére szatellitek állnak rendelkezésre.

Jelenleg kisérletek folynak az algoritmus gyakorlati megvalósitására a KSH IBM 370/155-ös OS/VS1 alatt üzemelő számitógépére.

Irodalom

[1] IBM TSS/360 System reference manual (C28-2003)

[2] W. J. Doherty: Scheduling TSS/360 for responsiveness (AFIPS Proceedings FJCC, 1970)

[3] Y. Bard: Experimental evaluation of System performance IBM Systems Journal (1973/3.)

[4] IBM OS/VS1 Planning and use guide (C24-5090)

[5] IBM 370 üzemeltetési szabályok KSH.

[6] UNIVAC Real-time Verarbeitung. (Sperry Rand Univac, Zürich, 300666/500)

Summary

Dynamic resource allocation in real-time and batch environment

Peter Gyarmati

The paper examines the general properties of dynamic allocation strategies and gives the description of a new algorithm.

Р Е З Ю М Е

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ РЕСУРСОВ В РЕАЛЬНОМ ВРЕМЕНИ И ПУЧКОВОЙ РАБОТЕ

П. Дярмати

В работе рассматриваются общие характеристики динамических стратегий распределения и дает новый алгоритм.

A konferenciát a "Számitástechnika tudományos kérdései" c.
többoldalú akadémia együttmüködés keretében rendeztük.

Конференция была проведена в рамках
многостороннего сотрудничества академий
социалистических стран по проблеме
"Научные вопросы вычислительной техники"

Conference was held in the frame of the multilateral cooperation
of the academies of sciences of the socialist countries on Computer Sciences.