Systémy reálného času. Měkké a tvrdé systémy. Plánování v systémech reálného času: plánování s periodickými úkoly, plánování založené na prioritách, přístup ke sdíleným zdrojům.

1 Systémy reálneho času

- Systém: procesor (aktívny), zdroj (pasívny), job (jednotka práce), task (opakujúci sa job)
- Hard deadline: requires verification, Soft deadline: requires validation.
- Periodic (pravidelný, hard), Aperiodic (nepravidelný, soft), Sporadic (nepravidelný, hard)
- Job - execution time e, release time r, deadline (relative/absolute) D/d. Response time: completion - release $C - r$, preemptible?, dependance/precendence?
- Schedule - dvojici job x čas priradí procesor a zdroje, scheduling sa deje len na racionálnych časoch.
- Ako riešiť závislosti: Podľa DAGu mením release/deadline tak aby release bol maximum z $r_i + e_i$ a môjho r a deadline minimum z $d_i - e_i$ a môjho d.

2 Plánovanie

- Obecne pre sadu jobov: Väčšinou stačí EDF, bez preemption alebo s resources už je to ale NP-hard
- Clock:
  - Viem že tasky sa mi zopakujú každú hyperperiod, takže stačí naplánovať hyperperiod.
  - Buď môžem mať presný scheduler (ťažké - clock drift etc.)
  - Alebo môžem mať periodické frames. Pričom teda platí že veľkosť framu delí hyperperiod, každý job sa zmestí do framu a medzi release a deadline musí byť aspoň jeden frame aby som mohol kontrolovať deadline overruns $2f - gcd(p_i, f) \leq D_i$.
  - Ak sú joby moc veľké, môžem ich umelo rezdeliť (preemptnúť na bezpečných miestach)
  - Aperiodic/Sporadic: Slack stealing - na koniec framu pridám a/s frontu. Vylepšenie: fronta ide na začiatok framu, ale potom to chce presné hodiny.
- utilization = e/p, total utilization = suma cez všetky tasky
- density = $e/min(p, D)$, prípadne pre sporadic je to $e/(d - r)$.

- Fixed priority:
  - Scheduling sa deje len na release/complete eventy.
  - Rate monotonic (menšia period = väčšia priorita), Deadline monotonic (menšia relative deadline = väčšia priorita)
  - Ak sú tásky simply periodic ($p_i > p_j \implies p_j \mid pi$), tak aj RM je optimálne (utilization 1).
  - Žiadny fixed priority algoritmus nie je lepší ako DM ak $D \leq p$ (Ak sa dá schedulenúť jedným algoritmom, určite sa dá aj cez DM).
  - Sada taskov je schedulable v DM ak má utilization at most $n \cdot (2^{1/n} - 1)$ - konverguje k $ln(2)$.
  - Time demand analysis: Počítam čas potrebný na vykonanie doteraz releasnutých jobov. Ak skutočný čas toto číslo nikdy neprekročí, mám problém.
  - Critical instant - job tasku ktorý má najväčší response time (ak phase = 0, tak sú to hneď prvé joby). Inak je to vtedy keď sú releasnuté všetky higher priority joby.
  - Test na schedulability: Time demand analysis v critical instant.
- Dynamic priority:
  - EDF / Least slack time
  - Sada taskov je schedulable v EDF ak má utilization at most 1.
  - Test na schedulability: Ak sú deadlines väčšie ako periódy, stačí testovať utilization, inak treba skontrolovať aj density (vtedy je to len necessary)
- Sporadic jobs:
  - Periodic server - normálny periodic task, ale má ešte aj budget ktorý môže minúť na sporadic/aperiodic tasks
  - Polling server - budget hneď zahadzuje ak nemá čo počítať
  - Deferrable server - budget si necháva, ale potom v najhoršom prípade môže zbrzdiť o 2x času.
  - Sporadic server - vždy keď prvý krát začnem pracovať, nastavím replenishment na teraz + perioda. Vylepšenie: Ak je neaktívny interval, toto nemusím robiť. Až skončí neaktívny interval, proste sa chovám ako keby nastal reštart vesmíru.
  - EDF: todo
- Resources:
  - Priority inversion: High priority job je blokovaný low priority jobom kvôli zdroju zatiaľ čo low priority je blokovaný medium priority.

– Primitívne: Job v kritickej sekcií má najvyššiu prioritu. Super lebo najdlhší blocking time je jedna sekcia, blbé lebo všetci ma môžu blokovať, aj keď sa o nič nebijeme.
– Priority-Inheritance - Ak job A drží resource ktorý chce iný job B s vyššou prioritou, job A dedí prioritu jobu B. Pekné: Nemôžu ma blokovať nesúvisiace joby. Blbé: Môžem byť blokovaný po dobu všetkých kritických sekcií v kolidujúcich joboch (Ale z každého len jednu).
– Priority-Ceiling - Prioritný strop zdroja je maximálna priorita jobov ktoré ho môžu potrebovať. Prioritný strop systému je strop maximálneho locknutého zdroja. Job môže dostať resource len ak má väčšiu prioritu ako súčasný priority ceiling, alebo rovnú, ale v tom prípade musí držať zdroj ktorý má tento strop. Dôsledok: Mám striktné usporiadanie na zdrojoch, takže nemám deadlocky a môžem byť blokovaný maximálne po dobu jednej kritickej sekcie. (Efektívne hovorím že zamknutím zdroja zakazujem konfliktné zamykanie všetkým menším jobom).