Besturingssystemen: Practicum

Simulatie van het geheugen en de processor bij paginatie

Vincent Naessens - 16 februari 2021

1 Doelstellingen

In dit practicum wordt de paginatiestrategie gesimuleerd en gevisualiseerd in Java, zoals gezien in de theorielessen. Het paginatiemechanisme wordt getest aan de hand van een XML-dataset die een lijst van instructies bevat met bijhorende parameters. Voor dit labo mag je in groepen van maximaal drie personen werken.

2 Eigenschappen van de hardware en het besturingssysteem

- Het **RAM geheugen** heeft de volgende eigenschappen. De reële adresruimte voor de code en de data van de processen bedraagt 48 Kbyte en is opgesplitst in 12 frames. Dit impliceert dat elke frame 4 KByte – of 2^{12} bytes – aan ruimte biedt voor de opslag van pages. Bijgevolg is de grootte van een page eveneens 4 KByte. Het RAM dat nodig is om delen van het besturingssysteem en page tables op te slaan wordt buiten beschouwing gelaten.
- De **virtuele adresruimte** is 2^{16} bytes. Dit betekent dat elk proces maximaal 64 KBytes groot is, en volledig op zowat elk hedendaags persistent opslagmedium kan opgeslagen worden. Elk virtueel adres bevindt zich in het interval [0, 65536-1]. Dit impliceert eveneens dat elk proces maximaal kan bestaan uit 16 pages, en bijgevolg dito entries moeten aangemaakt worden per page table (PT). Elke page ligt in het interval [0,15].
- Er wordt **1 page table (PT) bijgehouden per proces** door het besturingssysteem. Elke entry in de page table bevat (a) een present bit, (b) een modify bit, (c) de last access time, en (d) het corresponderende framenummer. Het framenummer ligt in het interval [0,11].
- In het RAM geheugen worden **maximaal 4 processen** toegelaten. Alle opgestarte processen moeten over evenveel frames beschikken. Elk proces krijgt dus maximaal 12 frames en minimaal 3 frames toegewezen. Indien een proces wordt opgestart worden frames gelijkmatig afgenomen van reeds aanwezige processen. LRU wordt hier gebruikt als globale vervangingsstrategie. Indien een page niet aanwezig is in het RAM van een instructie die moet worden uitgevoerd, dan wordt LRU eveneens als lokale vervangingsstrategie toegepast.
- De **aanwezige klok** is een integer die wordt opgehoogd telkens een operatie wordt uitgevoerd.

3 XML-dataset

Elke entry in de input dataset stelt 1 instructie voor en heeft de volgende structuur:
`<operation><pid>[<address>].`
Het `<address>` veld wordt enkel gebruikt bij `R` en `W` operaties (zie hieronder):

- `<operation>` : de operatie die door de processor moet uitgevoerd worden. Er zijn 4 mogelijkheden. `S` start een proces op; `T` beëindigt een proces; `R` leest de inhoud van een virtueel adres; `W` schrijft een andere waarde in de locatie van het virtueel adres.
- `<pid>` : de PID van het proces waarop de operatie wordt uitgevoerd.
- `<address>` : dit is de waarde van het virtueel adres. Dit is enkel belangrijk bij `R` en `W` operaties.

Merk op dat bij `S` een nieuwe page table wordt aangemaakt, en dat op dat moment bovendien pages frames in het RAM geheugen worden gealloceerd. Eventueel moeten frames van andere processen worden ingepikt. Bij `T` wordt de page table van het proces dat wordt afgesloten verwijderd, en worden frames herverdeeld onder de overblijvende processen.

4 Opdracht

De bedoeling van deze opdracht is de operaties (of instructies) in de XML dataset 1 na 1 uit te voeren, en de resultaten eveneens te visualiseren. De grafische user interface (GUI) bevat tenminste twee knoppen:

- **Optie 1.** Bij het indrukken van de **eerste knop** wordt telkens slechts 1 instructie uit de XML dataset uitgevoerd, en wordt de nieuwe toestand gevisualiseerd.
- **Optie 2.** Bij het indrukken van de **tweede knop** wordt de ganse XML dataset in 1 keer uitgevoerd. Er wordt bovendien weergegeven hoeveel keer een pagina werd geschreven van het persistent geheugen naar het RAM, en omgekeerd. Elke schrijfopdracht hier wordt veroorzaakt door een paginavervanging. Dit correspondeert dus niet met het tellen van de `W` operaties per proces!

Bovendien worden de volgende gegevens gevisualiseerd in de grafische user interface. Dit is voornamelijk belangrijk bij optie 1:

- De **timer**. Dit is een natuurlijk getal.
- De **instructie** die op dit moment wordt uitgevoerd door de computer. Dit is de eerstvolgende instructie die wordt opgehaald uit de XML dataset. Dit bevat ondermeer het virtueel adres van de instructie die net is uitgevoerd, en de instructie die als volgende zal worden uitgevoerd. Visualiseer eveneens het reële adres, in het bijzonder de frame en offset van dit adres.
- De **page table (PT)** van het proces dat op dit moment een instructie zal uitvoeren. Deze bevat 16 entries met verschillende data (zie vorige sectie).
- Een gedeeltelijke voorstelling van het **RAM**. De 12 frames worden weergegeven, samen met de `<pid>` en het pagenummer dat momenteel aanwezig is in dat frame.
- Het **reële adres** van de geheugenlocatie in het RAM die moet aangesproken worden.

Bovendien wordt er per proces bijgehouden hoeveel schrijfopdrachten van het persistent geheugen naar het RAM werden uitgevoerd, en vice versa (zie hierboven). Je software moet dus in staat zijn om bij elke `R` en `W` na te gaan of die uit te voeren instructie in het RAM aanwezig, en indien nodig paginaś te vervangen. Hiervoor wordt LRU gebruikt als vervangingsstrategie. Bovendien moeten de virtuele adressen vertaald worden naar reële adressen.

5 Evaluatie

Het geleverde werk wordt gequoteerd op 50 procent van de labopunten en bestaat uit 3 componenten:

- De kwaliteit van het rapport – maximaal 3 paginas – dat wordt ingediend, en focust op de belangrijkste technologische uitdagingen die moesten overwonnen worden.
- De kwaliteit van de code en de demonstratie tijdens de mondelinge presentatie, evenals de correctheid van de visualisatie.
- De mondelinge presentatie die wordt gegeven, en het beantwoorden van de vragen tijdens deze voorstelling.

6 Praktische informatie

Het rapport en de broncode mail je naar `vincent.naessens@cs.kuleuven.be` **uiterlijk 30 april 2021** om 8u des morgens.