TDIU11 – Föreläsning 7

Repetition/utblickar

Filip Strömbäck

LINKÖPING UNIVERSITY

Planering

Vecka	Föreläsning	Seminarie
4	Processer och trådar	—
5	Filsystem och lagring	Utmaning 1: Schemaläggning
6	Minneshantering	Rapport 1: Filsystem I
7	Virtuellt minne	Utmaning 2: Filsystem
8	—	Utmaning 3: Virtuellt minne
9	Säkerhet	Rapport 2: Filsystem II
10	Repetition/utblickar	Utmaning 4: Säkerhet
11	Tentaförberedelse	Utmaning 5: Repetition (tisdag)
12	(omtenta-p)	Rapport: Uppsamling

LINKÖPING UNIVERSITY

Virtualisering: Översikt

Mål: köra olika OS på samma hårdvara.
Anledningar:

- Isolation
- Bättre nyttjande av resurser
- Inkompatibel mjukvara
- ...

Finns olika typer:

- Emulera helt system
  - Hypervisor (QEMU, VMWare, VirtualBox, ...)
  - Typ-1, Typ-2, ...
- Emulera delar av systemet
  - Paravirtualisering
- Olika usermode, samma kernel
  - `chroot`, namespaces, containers, ...

Motiverande exempel

Jag vill...

1. ...köra program skrivna för ett annat operativsystem
2. ...testa att kompilera program med ny version av kompilator/bibliotek
3. ...köra program för en annan CPU-arkitektur
4. ...isolera grupper av program från varandra

Virtualisering: Emulera hela systemet

Idé: Vi kan skriva ett program som emulerar en dator (CPU + annan hårdvara)
Vi kallar detta för en *hypervisor*
Det finns två typer:

- Typ 1
- Typ 2

Virtualisering: Emulera hela systemet

Idé: Vi kan skriva ett program som emulerar en dator (CPU + annan hårdvara)
Vi kallar detta för en *hypervisor*
Det finns två typer:

- Typ 1
- Typ 2

Typ 2

Hypervisor är en process i värd-OS:

- CPU-kärnor för gäst-OS är trådar i värd-OS
- Disk i gäst-OS är filer i värd-OS
- ...

- Kernel-instruktioner är långsamma
- Paging måste emuleras
- Vi vill ha hårdvarustöd!

Virtualisering: Emulera hela systemet

Idé: Vi kan skriva ett program som emulerar en dator (CPU + annan hårdvara)
Vi kallar detta för en *hypervisor*
Det finns två typer:

- Typ 1
- Typ 2

Typ 1

Hypervisor är sitt eget "OS":

- Hypervisor schemalägger virtuella CPU för gäst-OS
- Hypervisor kontrollerar åtkomst till hårdvara

- Hypervisor får enkelt tillgång till hårdvarustöd (ring -1)
- Måste startas innan vanligt OS

Virtualisering: Paravirtualisering

Så här långt emulerar vi ”riktig” hårdvara:

$$\text{Gäst-OS} \Rightarrow \text{Emulerad hårdvara} \Rightarrow \text{Värd-OS} \Rightarrow \text{Riktig hårdvara}$$

Problem:

1. Gäst-OS översätter systemanrop till något hårdvaran förstår
2. Hypervisor översätter sedan från emulerad hårdvara till högnivå systemanrop till värd-OS
3. Värd-OS översätter sedan till något som den riktiga hårdvaran förstår

Virtualisering: Paravirtualisering

Så här långt emulerar vi ”riktig” hårdvara:

$$\text{Gäst-OS} \Rightarrow \text{Emulerad hårdvara} \Rightarrow \text{Värd-OS} \Rightarrow \text{Riktig hårdvara}$$

Paravirtualisering:

- Tillhandahåll speciell hårdvara med gränssnitt på hög nivå!

Problem: gäst-OS måste ha stöd för ny hårdvara.

Virtualisering: Emulera bara usermode

En ”extrem” form av paravirtualisering är att bara emulera usermode:

- Ingen kernel behövs — vår hypervisor emulerar hela kernel
- Vi kör program för olika CPU-typer, kernels i samma kernel

Exempel:

- QEMU: kan emulera en annan CPU-arkitektur
- Wine: emulerar Windows på Linux
- WSL1: emulerar Linux på Windows

Virtualisering: Isolering på kernelnivå

Ibland vill vi isolera program från varandra:

- Olika program antar olika saker om miljö (versioner av bibliotek, etc.)
- Vi vill inte att ett program ska råka förstöra för andra program

Linux (och andra UNIX-varianter) tillhandahåller:

- `chroot`
  - Grund för ex.vis `schroot` i Debian
- *namespaces* (se `man namespaces`)
  - Nätverk, filsystem, user, ...
  - Används exempelvis för *containers* (ex.vis Docker)

LINKÖPING UNIVERSITY

Kryptografi: Översikt

Det finns olika primitiver:

- (Kryptografiska) hashfunktioner
- Symmetrisk kryptering
- Strömchiffer
- Asymmetrisk kryptering

Kryptografi: Översikt

Det finns olika primitiver:

⇒ (Kryptografiska) hashfunktioner

- Symmetrisk kryptering
- Strömchiffer
- Asymmetrisk kryptering

Hashfunktioner

Vi har en funktion $h(x) = y$ med följande egenskaper:

- Om $x_1 = x_2$ så är $h(x_1) = h(x_2)$
- Det är svårt att utifrån y hitta ett x så att $h(x) = y$
- Notera: y kan ses som ett "unikt ID" för x
- Ibland vill vi att $h(x)$ är dyr (för lösenord)

Kryptografi: Översikt

Det finns olika primitiver:

- (Kryptografiska) hashfunktioner

$\Rightarrow$ Symmetrisk kryptering

- Strömchiffer
- Asymmetrisk kryptering

Symmetrisk kryptering

Vi har:

- En hemlig nyckel k
- $E(k, m) = e$ som krypterar m
- $D(k, e) = m$ som dekrypterar e
- Alltså: $D(k, E(k, m)) = m$

Kryptografi: Översikt

Det finns olika primitiver:

- (Kryptografiska) hashfunktioner
- Symmetrisk kryptering

⇒ Strömchiffer

- Asymmetrisk kryptering

Strömchiffer

Ett problem med symmetrisk kryptering är att samma meddelande blir samma krypterade text. Vi har därför strömchiffer!

- Vi genererar en ström av "slumpad" data från en nyckel
- $r_n = S(k, n)$ där n är ett heltal från 0 och framåt
- Sen kan båda sidor generera samma sekvens, och ex.vis kryptera/dekryptera med xor.

Kryptografi: Översikt

Det finns olika primitiver:

- (Kryptografiska) hashfunktioner
- Symmetrisk kryptering
- Strömchiffer

$\Rightarrow$ Asymmetrisk kryptering

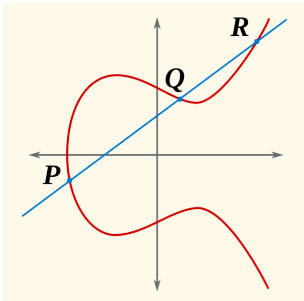
Asymmetrisk kryptering

Det är svårt att dela en nyckel mellan två eller flera parter...

- Två nycklar, en privat, k och en publik, K.
- $E(K, m) = e$ krypterar m
- $D(k, e) = m$ dekrypterar e
- Alltså: $D(k, E(K, m)) = m$
- Kan också *signera*: $D(k, m) = s$, då kan andra verifiera signaturen med $E(K, s) = m$

Kryptografi: Exempel: Elliptiska kurvor

- Kurva, *generatorpunkt* G
- Enligt bilden: $P + Q = R$
- Heltal $\Rightarrow$ punkt: $kP = P + P + \cdots + P$
- Nyckelpar: k privat, heltal, $K = kG$
- Generera delad hemlighet (avsändare):
  $r =$ slumptal, hemlig, $R = rG$, $S = rK$
- Hitta hemlighet (mottagare):
  $S = kR$, vilket är
  $S = kR = krG = rkG = rK$



Example elliptic curves, CC-SA, `https://commons.wikimedia.org/wiki/File:ECClines.svg`

LINKÖPING UNIVERSITY

Information

Denna delen av föreläsningen är upplagd i form av ett längre exempel som gås igenom på tavlan. Bilderna här agerar huvudsakligen som en sammanfattning av observationer och beräkningar. Motiveringen till dessa framgår alltså inte nödvändigtvis tydligt i bilderna. Informationen i bilderna bör dock vara tillräcklig för att kontrollera egna beräkningar.

Parametrar för vårt system

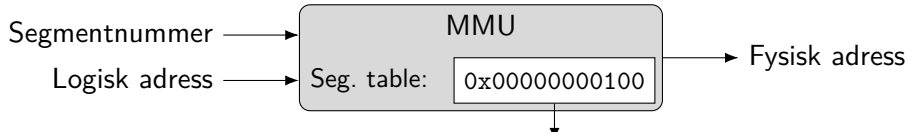
RAM:

- Virtuell adressrymd: 64 bitar, 48 bitar användbart (256 TiB)
- Fysisk adressrymd: 44 bitar (16 TiB)
- Accesstid: 10 ns

Disk (SSD):

- Fysisk blockstorlek: 1 KiB
- Storlek: 8 TiB
- Accesstid: 10 µs

Segmentering

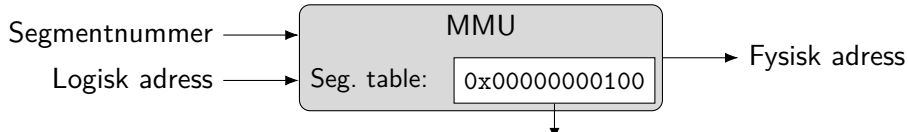


Seg	Base	Limit
0	0x00100000000	0x00180000000
1	0x00200000000	0x00280000000
2	0x00180000000	0x001A0000000

Hur många segment?

Hur stor blir segmenttabellen?

Segmentering



Seg	Base	Limit
0	0x00100000000	0x00180000000
1	0x00200000000	0x00280000000
2	0x00180000000	0x001A0000000

Hur många segment? Ex. första 16 bitarna för 2^{16} aktiva segment

Hur stor blir segmenttabellen? $2^{16} \cdot 2 \cdot 8 = 1$ MiB

Paging

Vad är en lämplig page-storlek?

- Stora pages $\Rightarrow$ mycket intern fragmentering
- Små pages $\Rightarrow$ många nivåer av page-tabell, hög overhead

Vi testar 4 KiB till att börja med

Paging – 4 KiB

- Page-storlek: 4 KiB, 12 bitar
- Hur stort behöver varje element vara?

- Hur många element får plats i page-tabell?

- Hur många nivåer behöver vi?

Paging – 4 KiB

- Page-storlek: 4 KiB, 12 bitar
- Hur stort behöver varje element vara?
  44 - 12 = 32 $\Rightarrow$ vi behöver plats för minst 1–2 bitar per element, 4 bytes/element fungerar inte, vi väljer 8 bytes/element (**dock mycket overhead**)
- Hur många element får plats i page-tabell?
  4 KiB / 8 bytes = 2^9 element
- Hur många nivåer behöver vi?
  4 nivåer:

9	9	9	9	12

Paging – 16 KiB

- Page-storlek: 16 KiB, 14 bitar
- Hur stor behöver varje rad vara?

- Hur många rader får plats i page-tabell?

- Hur många nivåer behöver vi?

Paging – 16 KiB

- Page-storlek: 16 KiB, 14 bitar
- Hur stor behöver varje rad vara?
  44 - 14 = 30 $\Rightarrow$ vi behöver plats för minst 1–2 bitar per rad, 4 bytes/rad går om vi klarar oss på 2 bitar/rad
- Hur många rader får plats i page-tabell?
  4 KiB / 4 bytes = 2^{10} rader
- Hur många nivåer behöver vi?
  3 nivåer:

10	12	12	14

Accesstid — Bara paging

Kom ihåg:

- Vi använder TLB som cache för uppslagning i page-tabell
- Accesstid, TLB: ε (försumbar)
- Accesstid, RAM: $t = 10$ ns
- Nivåer i page-tabell: $n = 3$
- Hit-ratio: $\alpha = 90\%$

$$\mathrm{EAT} = \alpha \cdot t_{good} + (1 - \alpha) \cdot t_{bad}$$

$$\mathrm{EAT} = \alpha \cdot (\varepsilon + t) + (1 - \alpha) \cdot (\varepsilon + (n + 1) \cdot t)$$

Accesstid — Bara paging

Kom ihåg:

- Vi använder TLB som cache för uppslagning i page-tabell
- Accesstid, TLB: ε (försumbar)
- Accesstid, RAM: $t = 10$ ns
- Nivåer i page-tabell: $n = 3$
- Hit-ratio: $\alpha = 90\%$

$$\begin{aligned}\text{EAT} &= \alpha \cdot t_{good} + (1 - \alpha) \cdot t_{bad} \\ \text{EAT} &= 0.9 \cdot (\varepsilon + 10 \text{ ns}) \\ &\quad + 0.1 \cdot (\varepsilon + 4 \cdot 10 \text{ ns}) \\ &= 13 \text{ ns}\end{aligned}$$

Accesstid — Virtuellt minne

- Accesstid, TLB: ε (försumbar)
- Accesstid, RAM: $t_r = 10$ ns
- Accesstid, Disk: $t_d = 10$ µs
- Nivåer i page-tabell: $n = 3$
- Hit-ratio, TLB: $\alpha_t = 90\%$
- Hit-ratio, RAM: $\alpha_r = 99\%$

Tänk: Räkna först EAT för *en* RAM-access ($\hat{t}_r$), sedan tar vi hänsyn till TLB:

$$\begin{aligned}\hat{t}_r &= \alpha_t \cdot (\varepsilon + t_r) \\ &\quad + (1 - \alpha_t) \cdot (\varepsilon + (n+1) \cdot t_r) \\ \mathrm{EAT} &= \alpha_r \cdot \hat{t}_r \\ &\quad + (1 - \alpha_r) \cdot (\hat{t}_r + t_d)\end{aligned}$$

Accesstid — Virtuellt minne

- Accesstid, TLB: ε (försumbar)
- Accesstid, RAM: $t_r = 10$ ns
- Accesstid, Disk: $t_d = 10$ µs
- Nivåer i page-tabell: $n = 3$
- Hit-ratio, TLB: $\alpha_t = 90\%$
- Hit-ratio, RAM: $\alpha_r = 99\%$

Tänk: Räkna först EAT för *en* RAM-access ($\hat{t}_r$), sedan tar vi hänsyn till TLB:

$$\begin{aligned}\hat{t}_r &= 0.9 \cdot 10 \text{ ns} \\ &\quad + 0.1 \cdot 4 \cdot 10 \text{ ns} = 13 \text{ ns} \\ \text{EAT} &= \alpha_r \cdot \hat{t}_r \\ &\quad + (1 - \alpha_r) \cdot (\hat{t}_r + t_d)\end{aligned}$$

Accesstid — Virtuellt minne

- Accesstid, TLB: ε (försumbar)
- Accesstid, RAM: $t_r = 10$ ns
- Accesstid, Disk: $t_d = 10$ µs
- Nivåer i page-tabell: $n = 3$
- Hit-ratio, TLB: $\alpha_t = 90\%$
- Hit-ratio, RAM: $\alpha_r = 99\%$

Tänk: Räkna först EAT för *en* RAM-access ($\hat{t}_r$), sedan tar vi hänsyn till TLB:

$$\begin{aligned}\hat{t}_r &= 0.9 \cdot 10 \text{ ns} \\ &\quad + 0.1 \cdot 3 \cdot 10 \text{ ns} = 13 \text{ ns} \\ \text{EAT} &= 0.99 \cdot 13 \text{ ns} \\ &\quad + 0.01 \cdot 10 \text{ µs} \approx 113 \text{ ns}\end{aligned}$$

Parametrar för vårt system

Disk (SSD):

- Fysisk blockstorlek: 1 KiB
- Storlek: 8 TiB
- Accesstid: 10 µs

Blockstorlek och storlek på filsystemet

1 KiB block ger:

- För 32-bitar blocknummer: $2^{32} \cdot 2^{10} = 2^{42} = 4$ TiB
- Vi har $2^{43}/2^{10} = 2^{33}$ block
- För stor disk för 1 KiB block och 32-bitars index

4 KiB block ger:

- För 32-bitar blocknummer: $2^{32} \cdot 2^{12} = 2^{44} = 16$ TiB
- Vi har $2^{43}/2^{12} = 2^{31}$ block
- Fungerar bra upp till 16 TiB.

Vi använder 4 KiB logiska block fortsättningsvis

FAT

Kom ihåg:

- Fungerar som länkad allokering
- Men, länkar samlade på ett ställe, i FAT
- Vill ha FAT i RAM för bra prestanda

Hur stor blir FAT för 8 TiB disk med 4 KiB logiska block, 32-bitars pekare?

FAT

Kom ihåg:

- Fungerar som länkad allokering
- Men, länkar samlade på ett ställe, i FAT
- Vill ha FAT i RAM för bra prestanda

Hur stor blir FAT för 8 TiB disk med 4 KiB logiska block, 32-bitars pekare?

- Antal block: $2^{43}/2^{12} = 2^{31}$
- Totalt: $2^{31} \cdot 2^2 = 2^{33}$ bytes $= 8$ GiB

Hur minskar vi storleken av FAT?
Vad är maximal filstorlek?

Indexerad, 1 nivå

Kom ihåg:

- Fungerar som 1 nivå paging
- Indexblock innehåller pekare till block med data

Vad är maximal filstorlek?
8 TiB disk, 4 KiB logiska block, 32-bitars pekare.

Indexerad, 1 nivå

Kom ihåg:

- Fungerar som 1 nivå paging
- Indexblock innehåller pekare till block med data

Vad är maximal filstorlek?
8 TiB disk, 4 KiB logiska block, 32-bitars pekare.

- Antal pekare i indexblock: $2^{12}/2^2 = 2^{10}$
- Maximal filstorlek: $2^{10} \cdot 2^{12} = 2^{22}$ bytes $= 4$ MiB

Hur ökar vi maximal filstorlek?

Indexerad, 2 nivåer

Vad är maximal filstorlek?

8 TiB disk, 4 KiB logiska block, 32-bitars pekare.

Indexerad, 2 nivåer

Vad är maximal filstorlek?

8 TiB disk, 4 KiB logiska block, 32-bitars pekare.

- Antal pekare i indexblock på 1:a nivå: $2^{12}/2^2 = 2^{10}$
- Antal pekare i indexblock på 2:a nivå: $2^{10} \cdot 2^{10} = 2^{20}$
- Maximal filstorlek: $2^{20} \cdot 2^{12} = 2^{32}$ bytes $= 4$ GiB

Hur stor plats tar indexblock för att lagra en fil som innehåller 10 bytes? För 5 MiB?

Filip Strömbäck

www.liu.se

LINKÖPING UNIVERSITY