Unikärnan

– ett alternativ till virtualisering och containrar

Av Ian Ferguson, Lynx Software Technologies

Ian Ferguson chefar över försäljning och marknadsföring på Lynx Software Technologies. Han tillbringade nära elva år på Arm, bland annat i ledningen för vertikal marknadsföring, företagsmarknadsföring och strategiska allianser. Ian Ferguson har en examen från Loughborough University (UK) med en kandidatexamen i elektro- och elektroteknik.

Unikärnor (unikernels) är ett försök att förena fördelarna med virtualisering och containerisering. De kan inte bara ge hög säkerhet utan även förbättrad prestanda och mindre fotavtryck.

Tekniken har faktiskt funnits åtminstone sedan 2016, men den har inte kommit i allmänt bruk. Inte ens i säkerhetskritiska tillämpningar, där den skulle kunna komma allra mest till sin rätt.

Vi inleder texten med med en genomgång av existerande arkitekturer för paketering av programkomponenter och jämför med de fördelar som unikärnor erbjuder – men även deras begränsningar.

Virtualisering vs containrar

Virtualisering – att köra multipla operativsystem på samma hårdvara – är en välutforskad teknik, om än inte särskilt effektiv när det gäller resursutnyttjande. Under några decennier var virtuella maskiner (VM) det som alla ville använda i sin serverinfrastruktur. På senare tid har industrin börjat gå över till att använda containrar istället, med system som Docker och Kubernetes.

Den ursprungliga arkitekturen för virtualisering var att låta varje VM kopiera upp en egen instans av ett komplett operativsystem. Det kräver duplicerade resurser och är dessutom en arkitektur som är svår att administrera med en mängd oberoende servrar.

DRIFTSÄTTNING AV PROGRAMVARA i en VM-arkitektur hanteras på ett av två sätt:

- ett byggverktyg producerar en komplett bootbar systemavbild för en VM där programvaran finns installerad. VM:en omstartas efter varje uppdatering.
- byggverktyget producerar enbart mjukvara och lägger den i ett paket som laddas upp till en server med hjälp av en samling skript.

BÅDA TILLVÄGAGÅNGSSÄTTEN kräver komplex installation och resulterar med tiden i att det växer fram inkonsistenser mellan VM:erna. Administratören har å andra sidan fullständig kontroll över systemets alla aspekter, och kan konfigurera så det passar specifika behov. Felsökning är relativt enkel att hantera genom att det går att koppla upp sig mot varje enskild VM.

Containrar, som Docker, är ett försök att återskapa idén med en VM-arkitektur, men eliminera dubbelarbetet.

Istället för att installera ett komplett operativsystem för varje app låter Docker dem köra i värdoperativsystemet, men dessutom ladda egna specifika bibliotek och program. Genom att justera containern och dess systemavbild (image) går det att exakt anpassa bibliotek och konfiguration för varje app. Detta ger ökad prestanda utan kostnaden för ett komplett operativsystem.

Containrar är lätta att köra på utvecklingssystem. Distributionsprocessen är även enklare eftersom man bara behöver ladda upp en redan kompilerad container till ett repository (kodförråd) och ett produktionssystem kan hämta hem uppdaterade versioner.

Containerarkitekturen har även sina nackdelar. Mjukvaran måste anpassas för användning i container, ”containeriseras”. Detta kan vara knepigt, särskilt med en äldre kodbas. Eftersom det finns många olika sätt att konfigurera en container vad gäller resursallokering och interoperabilitet, så är det ganska lätt att göra en felkonfigurering.

Unikärnor – fördelar och begränsningar

Nästa logiska steg i utvecklingen efter virtuella maskiner och containrar är unikärnor. De försöker driva konceptet med containrar ytterligare ett steg.

En unikärna är konkret en uppsättning förbyggda bibliotek i binärformat. En unikärna hanterar inte resursallokering och en hypervisor agerar mellanhand för interaktionen med hårdvaran. Hypervisorn ser till så att applikationsspecifika systemanrop flyttas så nära appen som möjligt.

Tanken med unikärnearkitekturen är att få den säkerhet som VM-partitionering ger och samtidigt den prestanda och det fotavtryck som en container ska ha.

Figur 1 visar skillnaden mellan de olika arkitekturerna.

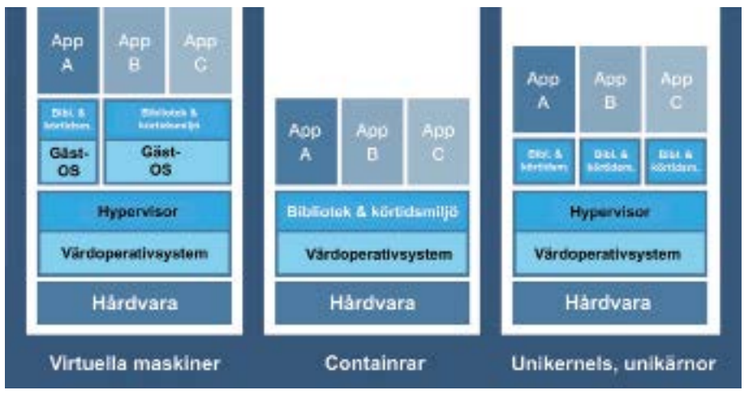
Där det krävs dataintegritet anser vi att man bör använda en typ 1-hypervisor (som saknar underliggande värdoperativsystem) som körs direkt på hårdvaran och laddas med VM:er. Detta eftersom ett underliggande värdoperativsystem utgör en sårbar punkt.

Unikärnor har ett ännu mindre overhead än en container, och är mer skräddarsydda vilket ska kunna ge bättre prestanda. Notera att här inte finns flera olika användare som delar på en gemensam kärna och adressrum. Detta förbättrar dramatiskt säkerheten.

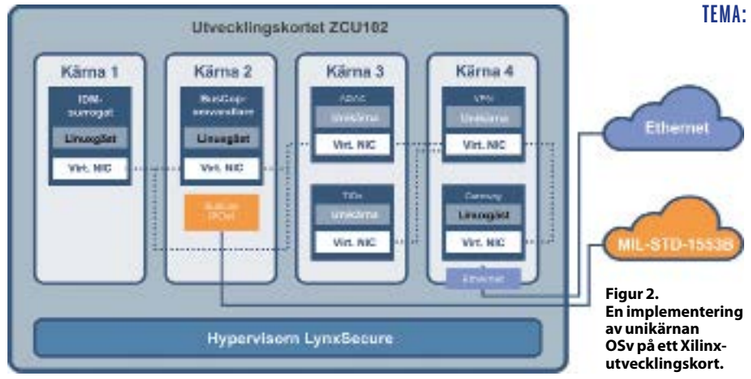
Operativsystem vs unikärna

En unikärna har klara säkerhetsfördelar mot ett operativsystem, men det finns också nackdelar.

En viktig fördel är att en unikärna till skillnad från ett operativsystem körs i en använ-



Figur 1. De tre arkitekturer som diskuteras här.



Figur 2. En implementering av unikärnan OSv på ett Xilinx-utvecklingskort.

dare och inte i kärnan. Det senare ska reserveras för funktioner på låg nivå med stora hårdvarurättigheter. En krasch i kärnan är en katastrof. Den stoppar hela systemet. Kod i användarläge har inte direkt åtkomst till hårdvara och minne.

Detta är faktiskt den grundläggande motivering vi hör från branschen, till att den börjat utforska unikärnor igen. Det går knappt en vecka utan en ny rapport om en cyberattack mot infrastruktur, företag eller myndigheter. Operativsystemet är en sårbar punkt.

En unikärna har även sina nackdelar. En av de viktigaste är att den bara hanterar en enda användare och en enda process.

Att addera processhantering innebär ett omfattande overhead med metoder för att starta, stoppa och inspektera processer, tillhandahålla interprocesskommunikation, etc.

Detsamma för fleranvändarstöd, som kräver autentisering, auktorisering, resursisolering, etc. Sådant slipper man i en enkel tillämpning med en enda funktion.

Det innebär tyvärr att vissa tillämpningar är näst intill omöjliga att implementera som unikärna. Inom Lynx fokusområde flyg och försvar, till exempel, skulle det vara en stor utmaning att försöka använda ett ARINC 653-bibliotek.

RÄTT SÄTT ATT HANTERA en sådan situation, enligt vårt förmenande, är

- att sprida ut systemet på enkla tillämpningar i varsin process.
- att installera ett operativsystem vid sidan av unikärnan, i en arkitektur som gör skadan av ett eventuellt intrång på operativsystemet minimal.

DET FINNS ÄVEN ANDRA PROBLEM som begränsat möjligheterna att ta unikärnor i bruk, fram till nu:

- Felsökning. Eftersom en unikärna inte kör något operativsystem, finns inget skal att koppla in sig på för att undersöka vad som pågår.
- Att producera en systemavbild av en unikärna är komplicerat och kräver djupa kunskaper i ämnet.
- De populära ramverk som används idag behöver anpassas till unikärnor och det måste dokumenteras hur de kan användas i unikärnor.
- Det saknas unikärnor som är certifierade för eller kan anpassas till verksamhetskritiska tillämpningar.

Så kan unikärnor börja användas inom fler områden

I början av 2022 gjorde Lynx en översikt över de olika alternativ till unikärnor som fanns allmänt tillgängliga. Målet var att försöka övervinna begränsningarna ovan, särskilt de sistnämnda.

Och dessutom att försöka hitta sätt att sprida fördelarna med unikärnor till ett bredare spektrum av tillämpningar.

Vi upptäckte att det fanns många unikärnor som var knutna till specifika kompilatorspråk: runtime.js och Javascript, Clive och Go, LING och Erlang samt Mirage och Ocaml.

Men vi hittade ett undantag, OSv, som kunde köra många olika runtimes. Därför valde vi att forska vidare i OSv, som är en modulär unikärna i öppen källkod.

Den kör ett ensamt Linuxprogram ovanpå en hypervisor, på ett säkert sätt. Programmet ska vara en omodifierad binärfil, antingen x86-64 eller Aarch64.

Så OSv är kort sagt en Linuxkompatibel unikärna. Den har adopterats i IT-infrastrukturen och har där utrustats med ganska bra verktygsstöd för transparens, som loggning, spårning och felsökning.

Den täcks av en lätthanterad BSD-licens och stödjer diverse protokoll och gränssnitt, som TCP/IPv4, PCIe och ACPI. Stödet för IPv6 var dock inte moget vid tiden för vår undersökning.

Unikärnor i verksamhetskritiska säkerhetstillämpningar

Vad gäller Lynx var vi primärt intresserade av unikärnor på grund av säkerhetsaspekten. De krymper angreppsytan drastiskt. Den här typen av tillämpningar ställer heller inga krav på tidsgarantier eller säkerhetscertifieringsartefakter.

Till en OSv-unikärna skulle man till exempel kunna delegera säkerhetskomponenter som intrångsdetektering (IDS) eller virtuella privatnät (VPN).

Figur 2 visar hur implementering på ett ett Xilinx-utvecklingskort skulle kunna se ut. Tillämpningarna är TIDS (Tactical Intrusion Detection System) och DAC (Dynamic Access Control). Det är statistiska anomalidetektorer som US Army använder för övervakning av IP- och 1553-trafik på några av sina nätverk.

Genom att addera enkelriktad dataöverföring (datadioder) och filter till en unikärna kan den ersätta en Linux-VM, vilket sparar minnesutrymme för kunden och drastiskt minskar angreppsytan.

En arkitekturmässig fördel, enligt oss, är att unikärnor minskar behovet av bare metal-kod – programvara som kommenderar hårdvaran direkt utan ett operativsystem som mellanhand. En unikärna kan tillhandahålla ett API (Lynx stöder exempelvis Posix) för att underlätta utvecklandet av tillämpningar.

Med detta sagt så finns det fortfarande plats för att låta enstaka funktioner hanteras bare metal. Även om unikärnor ger möjlighet att avsevärt minska programvarans minnesfotavtryck kommer det alltid att vara klart effektivare att implementera en applikation i 500 rader bare metal-kod. Kod som hanterar privata nycklar kan vara ett exempel.

Marknadsmognad

Är unikärnor redo för användning i verksamhetskritiska tillämpningar inom flyg och försvar?

Nja. Även om de har en del attraktiva fördelar finns fortfarande områden där det ställs strikta krav som dagens unikärnor inte riktigt lyckas uppfylla:

- Posix- och Arinc-kompatibilitet. Branschen övergår till att använda modulära arkitekturer med API:er som Posix (och FACE på den amerikanska marknaden, efterfrågad av US Army).
- Ett annat problem är storleken. Ju större ett operativsystem är, desto mer tid och pengar krävs för ta det igenom DO-178 och andra certifieringsprocesser. De striktaste versionerna av DO-178 kräver i snitt 2–4 timmars arbete per källkodsrad.
- Algoritmen för schemaläggning i OSv är preemptive och fair-share så att alla systemanvändare och grupper får använda processorn lika mycket. Preemptive betyder att processerna turas om att använda cpu:n under korta tidsintervall. Problemet är att det i verksamhetskritiska system förekommer att vissa uppgifter behöver förtur till resurser (det är inget jämlikt samhälle). Detta kräver en schemaläggning där det finns möjlighet att låta en process behålla cpu:n tills den är klar eller tar paus.

Slutsats

När det gäller verksamhetskritiska tillämpningar anser Lynx visserligen inte att unikärnor är fullt redo för allmänt bruk. Men den har fördelar som borde kunna övertyga en utvecklingsavdelning om att ta sig en funderare över om kanske inte kan finnas specifika tillämpningar där den passar.

Är du leverantör kan du hjälpa till genom att ta fram en lösning som integrerar unikärnor med klassiska realtidsoperativsystem. Så du behöver en unikärna som spelar bra ihop med sådana. ■