

# Mitme protsessori tugi operatsioonisüsteemides

Meelis Roos

ATI seminar

29. november 2001

## Riistvara

- Sümmeetriline vs asümmeetriline tööjaotus
- Ühine või eraldatud mälu
- Üks või mitu operatsioonisüsteemi koopiat

# SMP

- Sümmeetriline multiprotsessorsüsteem
- Ühine mälu ja S/V siin
- Mälu koherentsus
- Üks koopia operatsioonisüsteemist
- Sõltuvalt platvormist max 4 kuni 64 protsessorit
- Loetakse efektiivseks, kui viimasest protsessorist on kasu vähemalt 70 kuni 80 %
- Piduriks ühine mälu
- Vahemälul uus roll

## Alternatiivid

- ASMP – asümmeetriline multiprotsessorsüsteem
- Klastrid
- MPP – massiivselt paralleelne arvutamine

## Klastrid

- Palju masinaid ühendatud kiire võrguga
- Skaleeruvus protsessorite arvu poolest
- Veakindlus
- Hallatavus
- Ühe protsessori kohta väiksem töökiirus
- Näiteks VAXCluster, Beowulf

## MPP

- Massiivselt paralleelne süsteem
- Sama mälupilt kõigi protsessorite jaoks
- Igal protsessoril või väikesel protsessorite hulgal oma koopia operatsioonisüsteemist, oma mälu ja IO
- Kohalik mälu ja IO on kiire
- Kauge mälu on mitu korda aeglasem, aga kah nähtav
- Tuhandeid protsessoreid
- Näited: NUMA-Q, ccNUMA

## UNIX ühe protsessoriga arvutile

- Protsessidel on erinevad prioriteedid
- Scheduler (eesti keeles = ?) valib kõrgeima prioriteediga valmis oleva protsessi
- Protsess võib protsessoriaja ise käest ära anda (sleep(), blokeeruvad funktsioonid)
- Scheduler laseb protsessil joosta kuni tema ajakvant täis saab
- Protsesse ei vahetata, kui nad on parajasti tuuma kontekstis
- Katkestused katkestavad jooksva töö (isegi vähemprioriteetse katkestuse töötlemise)
- Kriitilistes sektsioonides blokeeritakse katkestusi: spl\*(), cli()  
jne

## Lihtne SMP

- Tuuma andmestruktuurid on kaitstud ainult sama protsessori katkestuste eest
- Mitme protsessoriga tekib probleem: ainult üks protsess saab tuuma moodis joosta
- Lahendus: SUUR LUKK (BGL, BKL, *funneling*, . . .)
- Kasutajamood skaleerub nüüd mitme protsessori peale
- Tuuma mood ei skaleeru, palju protsessoriaega läheb raisku

## Skaleeruv SMP

- *Fine-grained* lukustamine: lukustame ainult vajalikes kohtades
- Kui ressurss pole kättesaadav, blokeerume, mitte ei tee tühja tsüklit oodates
- Probleem: katkestuste teenindamine ei saa ju blokeeruda
- Lahendus: laseme tal siis blokeeruda

## Katkestused

- Anname katkestuste teenindusprogrammidele oma tuuma lõime konteksti (pinu+teenindusinfo) või midagi sarnast
- Edasi mitmeid võimalusi: schedulime kui harilikku protsessi/lõime, schedulime täiesti omaette sõltumata päris protsessidest, ...

## Probleeme

- Konkurents lukkude pärast (*contention*): kui lukud kehvasti paigutada, on alailma mõne luku taga keegi ootel
- Liiga palju lukke on kah paha, aur kulub vilele
- Larry McVoy: *locking cliff* – lukkudega on üle pingutatud, kui süsteemis on lihtsam lisada uus lukk kui välja mõelda, kas mõni olemasolev lukk piisav on. Viib allamäge.
- Cache sünkroniseerimine protsessorite vahel on kulukas. Kui sama andmestruktuuri eri protsessoritelt kasutada, kulub palju madala taseme võhma sünkroniseerimisele
- Üks lahendus on teha iga protsessori jaoks oma lokaalsed andmestruktuurid (näiteks nimekiri viimati kasutatud mälu plokkidest)
- Protsesside migreerimine protsessorite vahel on vahemälu tõttu keeruline

## Lukkude tüübid

- Semafor: fikseeritud arv juurde pääsejaid, ülejäänud on järjekorras ja blokeeruvad seniks
- Spin lock: ainult üks saab ligi, teised ootavad pidevalt uuesti proovides
- Blokeeruvad lukud (ootejärjekorrad): nagu spin lock, aga ootaja blokeerub
- Adaptiivne lukk: tsükeldab natuke aega, edasi blokeerub
- Lugemis-kirjutamislukk: laseb ligi ühe kirjutaja või mitu lugejat
- Tingimuslikud muutujad, . . .
- Lukud saavad olla rekursiivsed, aga tihti ei lubata seda

## Ilma lukkudeta läbi ajamine

- Atomaarsed muutujad
- Atomaarsed operatsioonid
- Lukustamist mittevajavad algoritmid
- Mälubarjäärid

## Solaris

- Skaleeruv SMP alates Solaris 2.0/2.1
- Preemptable tuum (eesti keeles = ?)
- Reaalaja tuum madala latentsiga
- Tuumas lõim iga kergprotsessi (*LWP*) jaoks
- Lisalõimed katkestuste, STREAMSi, NFS jms jaoks
- MT-safe ja mitte-MT-safe draiverid ja STREAMS moodulid
- Tänapäeval kuni 64 protsessorini skaleeruv

## Tru64 UNIX

- Skaleeruv SMP alates versioonist 3.0
- spl\*() ja *funneling* aegamööda minema
- Preemptable tuum
- Reaalaja tuum
- Tänapäeval vähemalt 32 protsessorini skaleeruv

## FreeBSD

- Versiooni 5.0 arendatakse skaleeruvat SMP-d (SMPng) Solarise ja BSDI eeskujul
- Big Giant Lock minema, *fine-grained* lukustamine asemele jõudumööda
- Katkestused omaette lõimedesse, spl\*() minema
- Kasutatakse põhiliselt blokeeruvaid lukke
- Katkestustel esialgu päris lõimed, lukustamise töökindlaks saamisel asendatakse (asendati?) kergemate lõimedega
- Skaleeruvus minule teadmata

# Linux

- Skaleeruv SMP alates versioonist 2.2
- Omamoodi disain, sest ei baseeru UNIXi ja BSD koodibaasil
- Tuuma tasemel 3 kihti: protsessikontekst, softirq, irq
- Katkestuste teenindamine käib kahes madalamas kihis
- Iga alumine kiht võib ülemisi katkestada
- Iga ülemine kiht saab alumise katkestamise vastu blokeeruda ja teiste protsessorite vastu luku võtta
- Kasutatakse palju spin locke, aga ka ootejärjekordi ja semafore
- Skaleeruvus: enamasti riistvaral 8 protsessorini, on nähtud töötamas kuni 32 protsessoriga serveritel (skaleeruvus teadmata)