,FSOFM JOUFM

0 .JOJY DPTUVNB TFS DPOTJEFSBEP P BOUFDFTTPS EP -JOVY NBT TFVT LFSOFMT TÍP CFN EJGFSFOUFT 0 OPWP .JOJY BHPSB DPN VNB MJDFOÎB OP FTUJMP #4% FTUÈ FN CVTDB EF VTVÈSJPT QPS 3àEJHFS 8FJTS

0 -JOVY QPTTVJ VENËBOS F MBÉÈBPPCKFUJWJEMEF RJVEBTMVPSPKÃ MPOHB F UFNQPNTUVCPTBPT TFS TVB DBMBOHUVFSHFOJDBEZUBSBEMU D VN PVUSP TJTUFENTBTJNQOPJGOVTY SFDPOFFGDFSVSBNFNQBS EF SFEF DPOIFDJEP DPNP0.JGOBÄYODJB EP USBCBMIBPSEVJUFOVSCBSVNTJTUFO NPTP BVUPS QSPGFTTPSBÉÈP JEFUBVEBJQYSÓQBOFTVNEFJBTFDV DPNQVUBÉÈP "OESFX&4NTVBBOCUPNPHSBOBSP .JOJY VN CPN DBO MBOÉPV B QSJNFJSBGWMSTNÈPEP BJJOJEYBOTÈFUFSNBEVFJNECBQBSBEPT FN èðïî DPNP VNB GFQSBNVÉTQBSJBOVT EBESFCVQFSBJPSMWSDPO FOTJOBS TJTUFNBT PEQFSBOEPCBVSNBJTFNQSZPNUFNTPS DPNP à SFXB BMVOPT 3BQJEBNFOWFTFFTDFBQEFRVNQMFNFOUBUJPO F CFN EPDVNFOUBEP EVTVFMFBRVEFONVEPV TVBOTJEB QPQVMBSJEBEF DPN PT EFDBVTJBTPCBF MJDSPLFSOFMT WFSTVT TJTUFNBT PQFSBDJPOBJSTOF/MNENEOPMÊUJDPT DQPUJOVB BUE PT HFN FOWJBEB BP HSVQPTEBFUEVBTJTVTÈEB NFTQSPCMFNBTRVF TFHV EP .JOJY P KPWFN VPJWFSTJOÃSJEFàBQBUSFDFV QFMJBOJYBGSB BU MBOEËT -JOVT 5PSWBMENÊNVOQSNBGFNDPQFSBTDFOUBJT" JODMVJ èððè TFV QSÓQSJP TJWUFSNTÈFYêQESJNFTUBNBPP-QFOSVYDJTÈOPBSMFEV DIBNBEP -JOVY 1PSUBOUFONBVJUPTBEPT QSPKFUP 0T FSSPT GPS QSJNFJSPT DPMBCPSBEPSFT EP -JOVY WJFFN TVB NBJPS QBSUF E SBN EB DPNVOJEBEF EP .JOJY USBJT EB EÊDBEB EF èð 1PSÊN 5BOFOCBVN F 5PSWBMET EFTTFT QSPCMFNBT QPE EFTEF DFEP DPNFÉBSBN B TF EJHMBEJBS BP GBUP EF RVF EFTFOW RVBOUP B BTQFDUPT EF QSPKFUP 1PS VNTÈP QFSGFJUPT)VNBOPT MBEP 5BOFOCBVN TFNQSF GBWPSFDFV SPT 0CWJBNFOUF TFSJ B BSRVJUFUVSB EF NJDSPLFSOFM VN PT OÙNFSPTF BCSBOEB SFDVSTP QBSUJDVMBS EP .JOJY OPT EJBTFOUSFUBOUP PT BSRVJ BUVBJTXWHKKBV-JOVST QPS NFOUF TF NPTUSBN EJTC PVUSP MBEP DSJPV P -JOVY DPNP VN SFDFS B WFMPDJEBEF FN LFSOFM NPOPMÎUJDP DPN TJTUFNBT EF TFHVSBOÉB F EB FàDJËO BSRVJWPT ESJWFSTWÃPSVJUPSTPQSDPKNFQUPTFOBOOFDYBMNTÊVBQNBPFTTF UFT JODPSQPSBEPTHBSBNBSiQVMNFNSNBG $PEF1BDUP 'BVTUJBOPu NFOTBHFN GBNPTB FOWJBEBYBÊHDSVQQPBUÎW'FMÊDNPENBQTRVDFTU×FT SF EP .JOJY P DSJBEPSTBEPSÉÈNCJEF F UBNCÊNPDQNSPKFUPT NPOP SFGFSJV BP -JOVY DWÃPSJiVN NÃSBVJEFBT WJÈSPUVBTJTUÎVDFMTVBQESPC QBTTP EF WPMUB BP ÊNPVTÉÈÎXSBSF BJUPFNBVÊBPTQ à BOUF SFTQPTUB EPSÊPVFNBSPSVDBWEJVÊNPXTJTUFNBB JOUFJSP 6N QSPGFTTPS Ê VNB EWÃSQJSJNÊUBEPTGDVÎÊVPEBNFOUBM Ê RVF PT

Quadro 1: Por que os computadores não funcionam sem parar?

Os usuários de computadores estão mudando. Há dez anos, a maioria dos usuários de computadores era pessoas ou profissionais jovens com amplo conhecimento técnico. Quando algo saía errado – o que ocorria com frequência – eles sabiam consertá-las. A maioria deles consegue consertar computadores tão bem quanto um nerd de computador padrão sabe consertar seu carro. O que eles querem mais do que qualquer outra coisa é que o computador funcione o tempo todo, sem interrupções ou falhas.

Muitos usuários comparam automaticamente seus computadores a suas televisões. Ambos estão repletos de componentes eletrônicos mágicos e possuem telas grandes. A maioria dos usuários tem um modelo implícito de uma televisão: (1) você compra a TV; (2) você a liga na tomada; (3) ela funciona perfeitamente sem qualquer falha durante os próximos dez anos. Eles esperam isso do computador e, quando não é o que obtêm, ficam frustrados. Quando os especialistas em computadores lhes dizem: "Se Deus quisesse que os computadores funcionassem o tempo todo, Ele não teria inventado o botão de RESET", eles não se convencem.

Por falta de uma melhor definição de disponibilidade, adotemos a seguinte: um dispositivo é dito disponível (isto é, podemos dispor dele) se 99% dos usuários jamais experimenta qualquer falha durante todo o período em que o possuem. Por essa definição, virtualmente nenhum computador é disponível, enquanto a maioria das TVs, iPods, câmeras digitais etc. são. Usuários técnicos de computador estão dispostos a perdoar um computador que trave uma ou duas vezes por ano; usuários comuns, não.

Usuários domésticos não são os únicos incomodados com a baixa disponibilidade dos computadores. Até mesmo em ambientes altamente técnicos, a baixa disponibilidade dos computadores é um problema. Empresas como Google e Amazon, com centenas de milhares de servidores, experimentam várias falhas todo dia. Elas aprenderam a conviver com isso, mas prefeririam sistemas que simplesmente funcionassem sem parar. Infelizmente, os softwares atuais falham nesse aspecto.

O problema básico é que softwares contêm bugs, e quanto mais software, mais bugs. Vários estudos já mostraram que o número de bugs por mil linhas de código (KLoC) varia de um a dez em grandes sistemas de produção. Um software muito bem escrito talvez tenha dois bugs por KLoC ao longo do tempo, mas não menos. Um sistema operacional com, digamos, 4 milhões de linhas de código, portanto, deve ter pelo menos 8 mil bugs. Nem todos são fatais, mas alguns serão. Um estudo da Universidade Stanford mostrou que drivers de dispositivos – que compõem até 70% da base de código de um sistema operacional típico – possuem taxas de bugs 3x a 7x mais altas que o resto do sistema. Drivers de dispositivos têm taxas mais altas porque (1) são mais complicados e (2) são menos inspecionados. Enquanto muitas pessoas estudam o escalonador, poucas verificam os drivers de impressoras.

(ZVS\sqV! RLYULSZ TLUVYLZ

A solução para esse problema é retirar código do kernel, onde o dano pode ser máximo, e colocá-lo em processos do espaço do usuário, nos quais bugs não conseguem causar falhas de sistema. É assim que o Minix 3 é projetado. O sistema Minix atual é o (segundo) sucessor do Minix original, que foi lançado originalmente em 1987 como sistema operacional educativo, mas desde então foi radicalmente revisado para se tornar um sistema altamente disponível e autorrecuperável. Segue uma breve descrição da arquitetura do Minix; há mais informações em XXX NJOJY . PSH

O Minix 3 é projetado para rodar o mínimo de código possível no modo do kernel, onde bugs podem facilmente ser fatais. Em vez de 3-4 milhões de linhas de código no kernel, o Minix 3 tem aproximadamente 5.000 linhas de código no kernel. Às vezes, kernels desse tamanho são chamados de microkernels. Eles lidam com gerenciamento de processos no baixo nível, escalonamento, interrupções e o relógio, além de fornecerem alguns serviços de baixo nível para componentes do espaço do usuário.

A maior parte do sistema operacional roda como uma coleção de drivers de dispositivos e servidores, cada um rodando como processo comum do espaço do usuário com privilégios restritos. Nenhum desses drives e servidores roda como superusuário ou equivalente. Eles não conseguem nem acessar dispositivos de I/O ou o hardware MMU diretamente. Precisam usar serviços do kernel para ler e escrever no hardware. A camada de processos rodando diretamente no modo de usuário acima do kernel consiste em drivers de dispositivos, com o driver de disco, o de Ethernet e de todos os outros rodando como processos separados protegidos pelo hardware MMU, para não conseguirem executar qualquer instrução privilegiada e nem lerem ou escreverem em locais de memória além dos seus próprios.

Acima da camada de drivers vem a de servidores, com um servidor de arquivos, um servidor de processos e outros. Os servidores fazem uso dos drivers assim como de serviço do kernel. Por exemplo, para ler um arquivo, um processo do usuário envia uma mensagem ao servidor de arquivos, que então envia uma mensagem para o driver de disco para buscar os blocos necessários. Quando o sistema de arquivos os tem em seu cache, ele chama o kernel para movê-los para o espaço de endereços do usuário.

Além desses servidores, há um outro servidor chamado "servidor de reincarnação". Ele é o pai de todos os processos de drivers e servidores e monitora seu comportamento. Se ele descobrir um processo que não esteja respondendo a pings, ele inicia uma nova cópia a partir do disco (exceto pelo driver do disco, que fica oculto na RAM). O sistema foi projetado para que muitos (mas não todos) os drivers e servidores críticos sejam automaticamente substituídos enquanto o sistema funciona, sem perturbar os processos de usuário em execução e sem nem notificar o usuário. Dessa forma, o sistema é autorrecuperável.

Quadro 1: Por que os computadores não funcionam sem parar? (continuação)

Para testar se essas ideias funcionam na prática, conduzimos os seguintes experimentos: iniciamos um processo de injeção de falhas que sobrescreveu 100 instruções de máquina no binário do driver Ethernet em execução para ver o que ocorreria caso um deles fosse executado. Se nada acontecesse em poucos segundos, outras 100 eram injetadas e assim por diante. No total, injetamos 800.000 falhas em cada um dos três diferentes drivers Ethernet e causamos 18.000 travamentos do driver. Em todos os casos, o driver foi automaticamente substituído pelo servidor de reencarnação. Apesar de injetar 2,4 milhões de falhas no sistema, o servidor não parou uma vez sequer. Nem é preciso dizer que se ocorrer um erro fatal num driver do Windows ou do Linux rodando no kernel, todo o sistema operacional travará imediatamente.

Existe alguma desvantagem nessa técnica? Sim. Há uma redução de desempenho. Não a medimos extensivamente, mas o grupo de pesquisa em Karlsruhe, Alemanha, que desenvolveu seu próprio microkernel (o -) e depois rodou o Linux como um de seus processos de usuário, conseguiu uma perda de desempenho de apenas 5%. Acreditamos que se dedicarmos um pouco de atenção a esse típico, também conseguiremos reduzir a perda para a faixa entre 5 e 10%. Desempenho não é uma prioridade para nós, já que a maioria dos usuários que leem email ou navegam pelo Facebook não são limitados pelo desempenho da CPU. O que eles querem, no entanto, é um sistema que simplesmente funcione o tempo todo.

:L TPJYVRLYULSZ ZqV [qV KPZWV UPUN\tT VZ \ZH&

Na verdade, usam, sim. Provavelmente você roda vários deles. Seu telefone celular, por exemplo, é um computador pequeno, mas comum em todos os outros aspectos, e há uma boa chance de ele rodar o L4 ou o Symbian, outro microkernel. O roteador de alta performance da Cisco também usa um microkernel. Nos mercados militar e aeroespacial, onde disponibilidade é fundamental, o Green Hills Integrity, outro microkernel, é amplamente usado. O PikeOS e o QNX também são microkernels amplamente usados em sistemas industriais e embarcados. Em outras palavras, quando é realmente importante que o sistema "simplesmente funcione o tempo todo", as pessoas usam microkernels. Para mais informações sobre esse tópico, veja XXX DT WV OM _BTU SFMJBCMF PT

Concluindo, é nossa crença, baseada em várias conversas com usuários não técnicos, que o que eles mais desejam é um sistema que funcione perfeitamente todo o tempo. Eles têm uma baixa tolerância a sistemas pouco confiáveis, mas atualmente não têm escolha. Acreditamos que sistemas baseados em microkernels podem nos levar a sistemas mais disponíveis.

DJPOBJT BUVBJT OÈP
EB NFOPS BVUPSJEBEF
FN JOHMËT &N SFTVNP
RVF PT EFTFOWPMWFE
CVJS PT TJTUFNBT Q
QBSB RVF VN FSSP F

Quadro 2: A questão da extensão

Muitos desenvolvedores e usuários discordam da doutrina de Tanenbaum, mantida há mais de uma década, de ser muito cauteloso quanto à introdução de extensões no kernel. O conceito de Tanenbaum a respeito da complexidade aceitável para o sistema operacional é um sistema que possa ser ensinado num único semestre. A modularidade permite que se complete o desenvolvimento de uma solução praticamente utilizável dentro do escopo de uma tese. Exemplos disso são os portes para várias arquiteturas de processador, modificações do Minix para a virtualização com Xen e aplicações de segurança.

Em sua autobiografia, Linus Torvalds revela seus motivos para rejeitar a arquitetura de microkernel para o Linux: "A teoria por trás de um microkernel sempre foi a separação do kernel em 50 partes independentes, e cada uma delas possui 1/50 da complexidade. Mas todos ignoram o fato de que a comunicação entre as partes na verdade é mais complicada do que o sistema original – sem contar que as partes ainda são não triviais". Um sistema monolítico desorganizado, portanto, pode oferecer benefícios de desempenho e escalabilidade, mesmo que lhe falte a estabilidade de um microkernel.

CJMJEBEF EF UPEP P TJ
EF DÓEJHP GFDIBEP FN
BNFBÉBN B TFHVSBOÉB
4FHVOEP 5BOFOCBVN J
LFSOFM VN ESJWFS GFD
BDFJUBS VN QBDPUF ME
FTUSBOIP F MFWÃ MP Æ
NBOEP EF VN BWJÈP

"SRVJUFUVSB
USBOTQBSFOU

0 .JOJY Ê QSPWBWFMNFO
PQFSBDJPOBM NBJT EPD
BUVBMJEBEF 0 MJWSP E
F 8PPEIVMM 5VFMMJOJY #PPL Ê
GFSËODJB QSJODJQBM
DBÉ×FT B SFTQFJUP EF O
QFTRVJTBT BUVBJT OB Q

< > 0 .JOJY TFHVF P QBESÈP 104*9
*&&& èççê é èðôí F PT EFTFOWPMWF
EPSFT KÃ QPSUBSBN WÃSJPT QSPHSBNBT
EP 6OJY QBSB FMF

%JGFSFOÎB

0 .JOJY ê FTUÃ OB FNFOUB EF WÃSJBT VOJWFSTJEBEFT F EJWFSTBT HFSBÉ×FT EF FTUVEBOUFT KÃ BOBMJTBSBN TFVT QPVDPT NJMIBSFT EF MJOIBT EF DÓEJHP F DPOTFS UBSBN B NBJPSJB EPT FSSPT " BSRVJUFUVSB EF NJDSPLFSOFM JNQMFNFOUB ESJWFST DPNP QSPDFTTPT TFQBSBEPT OP FTQBÉP EP VTVÃSJP RVF OÈP UËN QFSNJTTÈP EF FYFDVUBS DPNBOEPT QSJWJMFHJBEPT PV SFBMJ[BS PQFSBÉ×FT EF * 0 PV BJOEB EF HSBWBS EJSFUBNFOUF OB NFNÓSJB &N WF[EJTTP FTTBT PQFSBÉ×FT TÈP SFBMJ[BEBT QPS DIBNBEBT EF TJTUFNB

HPSB RVF FMF
DÓEJHP
F àSFXBMM
-TRVJTB
EB 5'') #FSMJO UBNCÉ
F`E% BàSRVJ
Qàsè NàSWFX BEJUs FSVXB@
VN BHSFTTPS RVF
PDBS VN USBWBNFOUP
FYQMPSBOEP VN FT
OB GVOÉÈP
P àSFXBMM -JOVY
QSPDFTTP EF
DPNQSPNFUFS B
0 TFSWJEPS EF
TJNQMFTNFOUF
UPSOBN BJOEB
QBSFOUFT TF P BHSFTTPS DPOTF
HVJS FYFDVUBS BMHVN DÓEJHP /P .J
JOJY
P TFRVFTUSP EF VN QSPDFTTP EF
QSPCMFNÃUJDP NBT
P TJTUFNB Ê NFOPT
FYDF TJTPMBNFOUP
B .JDSPTPGU FTUÃ FYQMPSBOEP
-JOVY WÃSJBT GBMIBT EF NJDSPLFSOFM
TVSHJSBN 4F VN TV
EF P .JOJY KÃ
UJQP FTUJWFS FN FY
F FP NJDSPLFSOFM
-JOVY FMF DPMPDBS
NBJPS PCTUÃDVMP Æ
SBÓÉB EP TJTUFNB BNQMB EJGVTÈP TFNQSF GPJ TVB MJDFOÉB