Інтэграцыя ў QEMU для віртуалізаваных старых аперацыйных сістэм

Дзягцярык І.А., Касцюк Д.А.

Брэсцкі дзяржаўны тэхнічны універсітэт
degtyarik.ilya@gmail.com

Running old-time OS in a virtual machine brings several problems, such as low computer architecture compatibility and bad user experience. Architecture compatibility is good with dedicated emulators, but they do not provide snapshots and other useful features of such mainstream virtualization systems as QEMU. So nested virtualization combining QEMU and some dedicated emulator is helpful in this case. When ancient hardware is supported by QEMU natively, you still have to solve the problem of mouse integration, also known as the problem of non-coinciding cursors.

Калі вы хочаце працаваць з віртуалізаванай старой аперацыйнай сістэмай, то першая праблема з якой вы сутыкаецеся - знайсці эмулятар, які будзе здольны яе запусціць. Другой праблемай звычайна становіцца дрэны карыстацкі вопыт пры выкарыстанні такой сістэмы ў эмуляцыі.

У прыватнасці, вы можаце ўбачыць высокую загружанасць працэсара, бо старыя аперацыйныя сістэмы не мелі магчымасці працы з цыкламі прастою працэсара, або з-за дрэннай аптымізацыі кода эмулятара.

Таксама можа здарыцца, што гасцёўная сістэма не будзе мець доступу да хост-сістэмы. У такім выпадку не будзе якіх-небудзь агульных дырэкторый для перадачы інфармацыі паміж машынамі, не будзе магчымасці перадаць файлы праз буфер абмену або USB.

Таксама графіка без апаратнага паскарэння ў старых сістэмах можа апынуцца праблемай, роўна як і адсутныя драйвера для сеткі, гуку, дыскавых сховішчаў і г.д. Нарэшце, практычна заўсёды ёсць праблема з незразумелымі рухамі мышы або яе блакаваннем.

Калі вы працуеце з састарэлай АС, у вас ёсць два шляхі: выкарыстоўваць якую-небудзь папулярную сістэму віртуалізацыі ці ж выкарыстоўваць спецыялізаваныя эмулятары.

Добрым прыкладам папулярнай сістэмы віртуалізацыі з'яўляецца QEMU. У яе ёсць магчымасць эмуляцыі вялікага спектру апаратных сістэм: x86, ARM, MIPS, Motorola 68K і г.д. Але на практыцы калі вы выбіраеце экзатычны працэсар у QEMU, вы не маеце паўнавартасны стары кампутар, а толькі сімуляцыю працэсара і некаторых спадарожных прылад - зусім не ўсёй перыферыі ранейшых гадоў. Таму гасцёўная аперацыйная сістэма можа проста не зарабіць. Таксама ў такім выпадку ёсць вялікія праблемы з інтэграцыяй мышы, што робіць карыстацкі досвед не занадта прыемным.

Асобныя эмулятары могуць быць трохі лепш, калі глядзець з боку вышэйпералічаных фактараў. Сярод іх ёсць як эмулятары з адкрытым

зыходным кодам, так і прапрыетарныя эмулятары са старых SDK. Праблем сумяшчальнасці ў іх адчувальна менш, бо яны напісаныя пад канкрэтную аперацыйную сістэму. Таму няма неабходнасці ў дадатковых драйверах. Але ёсць іншая сур'ёзная праблема. Такія эмулятары не маюць магчымасці захавання стану віртуальнай машыны (т.зв. snapshots, здымкі стану).

Таму часам даводзіцца звяртацца да укладзенай віртуалізацыі. У такім выпадку QEMU дае вялікія магчымасці. QEMU запускае віртуальную машыну x86 і нейкую «ўнутраную» аперацыйную сістэму, а тая запускае эмулятар, у якім у сваю чаргу і пачынае працаваць гасцёўная сістэма. Такія «ўнутраныя сістэмы» называюцца сэрвіснымі. Добрымі прыкладамі такіх сэрвісных сістэм з'яўляюцца, напрыклад, Linux, FreeDOS, ReactOS.

Здавалася б, накладныя выдаткі на укладзеную віртуалізацыю розных працэсарных архітэктур вельмі вялікія. Чаму ж такая укладзеная сістэма віртуалізацыі працаздольная? Усё проста. Звычайна састарэлыя сістэмы маюць вельмі малыя апаратныя патрабаванні да кампутара. Ім патрэбен вельмі маленькі аб'ём аператыўнай памяці і спажыванне CPU, адносна магутнасцяў сучасных машын.

Трэба сказаць, што многія састарэлыя аперацыйныя сістэмы могуць запускацца як звычайныя, без укладзенай віртуалізацыі. Напрыклад: Windows 1, 2, 3, 95, 2000, Pen Windows, Gem, Maemo, Android, WebOS.

Некаторыя састарэлыя АС маюць уласныя спецыялізаваныя эмулятары з адкрытым зыходным кодам. Вось іх кароткі спіс: Xerox Alto, GEOS, Amiga, RiscOS, Apple Lisa, MacOS 1, MacOS X, MacOS 7, і нават Apple Newton. Прапрыетарныя эмулятары таксама цалкам могуць быць карысныя. Іх прыклады - Xerox GlobalView, Psion EPOC16, EPOC32, PalmOS, Magic Cap, Windows CE.

Нарэшце, вернемся да праблемы інтэграцыі мышы. Калі мы выкарыстоўваем т.зв. «натыўную» віртуалізацыю, мы можам сустрэцца з рэжымам блакавання мышы, або нам прыйдзецца разбірацца з правільнай інтэграцыяй мышы ў сістэму. Калі ж мы выкарыстоўваем укладзеную віртуалізацыю, можа пашчасціць ня сутыкнуцца з гэтымі праблемамі, бо сэрвісная АС сама павінна перадаваць правільныя каардынаты паказальніка ў гасцявую сістэму пад ёй.

Праблема інтэграцыі мышы наступная. Гасцёўны курсор і курсор хост-сістэмы маюць розныя паскарэнне і хуткасць. Таму рух мышы прыводзіць да таго, што курсоры гасцёўнай сістэмы і хост-сістэмы маюць выдатныя адзін ад аднаго каардынаты. Звычайна сістэмы віртуалізацыі хаваюць курсор хост-сістэмы, і блякуюць кіраванне мышшу ў хост-сістэме, пакуль карыстач не націсне на клавіятуры адмысловую клавішу. Гэта называецца «mouse lock mode» - «захоп мышы». Праблема гэтага рэжыму відавочная: вы не можаце ўзаемадзейнічаць са сваёй галоўнай сістэмай, пакуль захоп мышы працуе. Аднак, калі вы карыстаецеся выдаленым доступам да гасцёўнай сістэмы, вы не зможаце выкарыстоўваць рэжым блакіроўкі мышы. У вас будуць

адлюстроўвацца два курсора, якія рухаюцца з рознай хуткасцю, і як толькі хост-курсор пакіне вобласць акна эмулятара, гасцёўны курсор перастане рухацца, так як віртуальная машына страціць фокус.

Як можна забяспечыць інтэграцыю мышы (прывесці курсоры ў адно палажэнне без блакавання)? Першы шлях досыць лагічны. Для дасягнення аднолькавых пазіцый можна эмуляваць ў гасцявой сістэме прыладу пазіцыянавання з абсалютнымі каардынатамі (напрыклад, планшэт). Растлумачым адрозненні ў метадах пазіцыянавання курсора. Пры адносным пазіцыянаванні (яго выкарыстоўвае мыш) для перамяшчэння паказальніка выкарыстоўваюцца «дэльты», г.зн. вектара перамяшчэння. Пры абсалютным пазіцыянаванні драйвер прылады генеруе дакладныя каардынаты для перамяшчэння паказальніка. Але для такой прылады абавязкова патрэбен драйвер ў гасцёўнай АС.

Другі шлях - перадаваць драйверу мышы гасцёўнай сістэмы дадатковую інфармацыю аб курсоры хост-машыны. У такім разе патрэбны спецыяльны драйвер ад пастаўшчыка сістэмы віртуалізацыі для КОЖНАЙ састарэлай аперацыйнай сістэмы, якую мы хочам запусціць. Улічваючы, што праграмаванне пад такія АС становіцца задачай лічбавай археалогіі, гэты падыход для запуску састарэлых сістэм вельмі дрэнны.

Як працуе інтэграцыя мышы ў QEMU? Для гэтага незаўважна ад карыстальніка для перамяшчэння курсора эмулюецца USB-планшэт Wacom PenPartner. Ён вельмі проста наладжваецца, паколькі выкарыстоўвае распаўсюджаны USB-інтэрфейс, і драйвера адразу прысутнічаюць практычна ва ўсіх USB-сумяшчальных гасцёўных АС.

Асноўнай праблемай такога падыходу з'яўляецца немагчымасць працы з сістэмамі, у якіх няма падтрымкі USB. Для вырашэння праблемы немагчымасці падлучэння па USB ёсць магчымасць эмуляцыі планшэта Wacom з паслядоўным інтэрфейсам. Ён выкарыстоўвае пратакол Wacom IV як і больш раннія прылады. Такі планшэт падтрымліваецца QEMU пачынаючы з версіі 1.9. Аднак для такога планшэта гасцёўныя аперацыйныя сістэмы маюць патрэбу ў дадатковай ўстаноўцы драйвера. Для некаторых сістэм, як, напрыклад, DOS, OS/2, Windows 3.x, Windows 95, BeOS, MacOS 9, MacOS X, AmigaOS, драйверы існуюць і могуць быць устаноўлены як мінімум для архітэктур x86 і x86_64.

Сумяшчальнасць рэалізаванага ў QEMU планшэта з паслядоўным інтэрфейсам сёння выглядае так. Для Windows 3.x і Windows 95 драйверы працуюць добра. Для DOS-драйвера патрабуецца невялікі патч. Драйвер для BeOS працуе часткова. Драйвер OS/2 не знаходзіць планшэт. Версія QEMU для архітэктуры PowerPC проста не мае паслядоўнага порта.

Сачыць за статусам падтрымкі інтэграцыі мышы ў QEMU для састарэлых аперацыйных сістэм і знайсці асаблівасці іх налады можна на старонцы http://ostimeline.org/wctablet.