Osprey

Noah "Sape" Evans

The world is changing

original Plan9

- Timesharing from commodity
- small
- statically administered
- low bandwidth terminals
- closely coupled file and cpu servers
- everything is a file/9p everywhere
- strong security model

World today

- Client/server interaction based on HTTP
- Poor, ad-hoc security
- Many layers of gratuitous protocol
  - REST over Websockets over HTTP over TCP over IP
- No model for accessing replicated static content
- Not to mention dynamic content

World tomorrow

- Large elastic “clouds” of machines
- Machines and applications are mobile
- Lots of replicated and distributed “content”
- Mobile applications supported by network services

Modern problems

- authentication and access control of network content and services
  - http has no built in auth mechanisms other than basic
- http not efficient
  - reloading a bunch of stuff over tcp
- constant mobility
- “connection” equated to “session”
- connections just aren’t reliable in a modern world

Plan 9 solves some problems

- Well authenticated RPC protocol(9p)
- Name spaces allow location-independent access

But not all

- I want to...
  - dynamically add xen systems to my NDB.
  - have my auth server proxy auth for me
  - make my system come to me.
    - data computation etc..
  - keep my state after drawterm crashes.
  - manage large-scale distributed applications.
  - manage dynamic, replicated data.

Solving these problems

Osprey

Osprey, what

- Authenticated, controlled access to dynamic replicated data
- support for cloud applications: remote invocation, checkpointing, migration
- good for new multicore processors
- good for hard real-time embedded development
- support for mobility and less than fully reliable wireless communication

Osprey, how

- microkernel w/library oses on top
- distributed delegated auth
- new session based file protocol, good for streaming and persistence
- migration possible

microkernel

- minimal state in the kernel only:
  - fast message passing
  - address space maintenance
  - interrupt handling
  - process scheduling

fast message passing

- inter-core message passing
- page flipping where possible

Internals

- address spaces
- segments can be shared different addresses
- processes share segments
- processes that to share address spaces migrate as a group

```
static void
uarttask(void *arg)
{
	char c;
	Uart *uart = arg;
	Task *ut = m->task;
	Msg *msg;
	Select sels[4] = {
		{ .op = MSget,    { .q = ut },           { .p = &msg }   },
		{ .op = MSdown, { .s = uart->sem },     { .b = nil }    },
		{ .op = MSget,    { .q = uart->qout },    { .b = &c }    },
		{ .op = MSlast,   { .q = nil },            { .b = nil }   },
	};
	int n;

	/* Starts when interrupts are enabled */
	while(uart->enabled){
		/*
		 * don't receive characters from output queue unless
		 * there's room in the fifo:
		 */
		sels[2].op = uart->sendrdy ? MSget : MSnoop;
		/* wait for an event: */
		n = select(sels);
		/* prevent interrupts, this IS a device driver: */
		ilock(uart);
		switch(n){
		default:
			_assert("i8250task: select");
		case 0:
			/* process message */
			uartmsg(uart, msg);
		case 1:
			/* process an interrupt */
			uart->phys->interrupt(uart);
			break;
		case 2:
			/* print character from output queue */
			uart->phys->sendc(uart, c);
			break;
		}
		iunlock(uart);
	}
	iprint("i8250task exits\n");
}
```

πp

- similar to 9p but sessions are now independent from connections
  - ie. if your connection dies your session stays alive if you want it to
- transactions can be grouped "pipelined"
  - ie, file can be opened, read and closed in one round trip.
- can be its own transport protocol
- support for isochronous data

the more things stay the same

- authentication
- fids
- T and R messages

the more they change

- string attributes
- versioning
  - all files are versioned
  - immutable, committed on file close
- file are leased
- clients can renegotiate session

```
uart       2.535s  Prepare topen
devroot    2.553s  parse group
devroot    2.555s  session: sid 12345678, tag 4
devroot    2.558s  topen, fid 1, nfid 2, name dev/uart/eia0, how r
devroot    2.564s  topen, chan /, fref 0x0/0x0
devroot    2.567s  topen, cloned fid
devroot    2.567s  topen, walk /dev/uart/eia0
devroot    2.567s  topen, dev->walk / to dev
devroot    2.576s  rootwalk / (0/0/0) → dev
devroot    2.578s  devwalk /dev (0x200 0/2/0)
devroot    2.585s  topen, path /dev
devroot    2.587s  topen, dev->walk /dev to uart
devroot    2.587s  rootwalk /dev (0/2/0) → uart
devroot    2.587s  devwalk /dev → uart
devroot    2.594s  devwalk /dev/uart (0x2 2/0/0)
devroot    2.600s  topen, path /dev/uart
devroot    2.600s  topen, dev->walk /dev/uart to eia0
devroot    2.605s  uartwalk /dev/uart → eia0
devroot    2.732s  uartdiscover: uarts 0xffffffffc0230a20, uartdir 0xffffffffc0230a68
devroot    2.737s  uartdiscover: 2 entries
devroot    2.739s  devwalk /dev/uart → eia0
devroot    2.741s  devwalk /dev/uart/eia0 (0x10002 2/0/1)
devroot    2.741s  uartwalk /dev/uart/eia0 == uart 0, COM1 @ 0xffffffffc012c6d0
devroot    2.741s  topen, path /dev/uart/eia0
devroot    2.750s  uartopen /dev/uart/eia0 r
devroot    2.753s  uartopen /dev/uart/eia0: COM1
devroot    2.755s  devopen /dev/uart/eia0 r 0xffffffffc0230a68
devroot    2.757s  session: reply 24 bytes
uart       2.757s  Got reply
uart       2.757s  ropen: done
uart       2.764s  Prepare tread
devroot    2.781s  parse group
devroot    2.781s  session: sid 12345678, tag 5
devroot    2.789s  session: read: preset rep->data 0xffffffffc022d454, 8204
devroot    2.794s  uartread
devroot    2.794s  uartread 256 characters attribute *
devroot    2.799s  uartread wait for response
Uarttask   2.801s  uartmsg get attr *
devroot    2.803s  session: reply 77 bytes
uart       2.805s  Got reply
rread attr=*: 65 bytes: '# * b c d e f h i k l m p r s x framing overruns berr cts dsr dcd'
uart       2.805s  Prepare tread
devroot    2.828s  parse group
devroot    2.830s  session: sid 12345678, tag 6
devroot    2.837s  session: read: preset rep->data 0xffffffffc022d454, 8204
devroot    2.837s  uartread
devroot    2.837s  uartread 256 characters attribute b
Uarttask   2.846s  uartmsg get attr b
devroot    2.837s  uartread wait for response
devroot    2.896s  session: reply 16 bytes
uart       2.896s  Got reply
rread attr=b: 4 bytes: '9600'
10 seconds type ahead: Uarttask    2.946s  uartmsg set attr d
aSDASDASD
uart       21.014s  Prepare tread
```

Applications

- High throughput batch jobs
- routing, gateways, firewalls
- Cloud OS
- Thin client OS
- HPC OS

Progress

- boots
- real time scheduler
- working on a packet filter
- πp devices in the kernel

Conclusions

- Plan9 is still the right engineering model
  - fs based way of unifying dist system
- but the current implementation is too static
- Osprey gets around these problems by using a microkernel with migratable processes and a caching filesystem

Demo

duvel (10.12.0.67!67): /amd64/kpc64
109969+28912+77552=216433
warp64(0xfffffffc0110000) 0xfffffffc0000000 16

This is Osprey

Ptr size 64
2666MHz 2666666667Hz 374999fs
pml4 0xfffffffc010a000
memstart 0x145000 2863104
level 4 page table for va 0x0000000000000000 at 0xfffffffc010a000
pte[511] va 0xffffff8000000000 → pa 0x000000000010b000 xi ac sw indirect
level 3 page table for va 0xffffff8000000000 at 0xfffffffc010b000
pte[511] va 0xfffffffc0000000 → pa 0x000000000010c000 xi ac sw indirect
level 2 page table for va 0xfffffffc0000000 at 0xfffffffc010c000
pte[0] va 0xfffffffc0000000 → pa 0x0000000000000000 x dac sw size 0x200000
pte[1] va 0xfffffffc0200000 → pa 0x0000000000200000 x dac sw size 0x200000
pte[127] va 0xfffffffcfe00000 → pa 0x000000000010d000 xi c sw indirect
level 1 page table for va 0xfffffffcfe00000 at 0xfffffffc010d000
ELCR: 0E00
pit0: hz 1193182 max 50000000000000 min 100000000000 mul 5120 58207
Enable asynchronous printing
dumptask: task Acedia at 0xfffffffc0186000
stack: 0xfffffffc0188000-0xfffffffc0188000
pi pc: 0x0, sp: 0x0
n messages received: 0
messages in queue: 0
gp periods: 20000409
on 3.766s best-effort time used
dumptask: task Uarttask at 0xfffffffc0188000
stack: 0xfffffffc018a000-0xfffffffc018a000
g pc: 0xfffffffc0123a1f, sp: 0xfffffffc0189ea8
: state Ready, scheduler patientia, events 0x0, flags 0x0
messages received: 0
messages in queue: 0
10 periods: 10
00 164.538µs best-effort time used
dumptask: task Ira at 0xfffffffc018a000
00 stack: 0xfffffffc018c000-0xfffffffc018c000
pc: 0xfffffffc01234de, sp: 0xfffffffc018be78
state Waitsem, scheduler patientia, events 0x0, flags 0x0
00 messages received: 0
p messages in queue: 0
in periods: 1
419.999ns best-effort time used
gdumptask: task pingpong at 0xfffffffc018c000
stack: 0xfffffffc018e000-0xfffffffc018e000
po pc: 0xfffffffc011eace, sp: 0xfffffffc018dba0
state Running, scheduler patientia, events 0x0, flags 0x0
ng messages received: 10000001
messages in queue: 0
s periods: 10000228
i 3.480s best-effort time used
dumptask: task Secondo at 0xfffffffc018e000
stack: 0xfffffffc0190000-0xfffffffc0190000
n pc: 0xfffffffc01234de, sp: 0xfffffffc018fe90
state Waitsem, scheduler patientia, events 0x0, flags 0x0
5 messages received: 10000000
messages in queue: 0
. periods: 10000196
40 1.954s best-effort time used
20000456 context switches
6 60001910 calls to now()
s 1105 timer interrupts

This was Osprey

msgsend+msgrecv+context switch: 270.309ns

```
duvel (10.12.0.67!67): /386/kpc32
108011+24180+40752=172943
entry: 0xf0110000
warp9(0x110000) 11

This is Osprey

pointer size 32
999MHz 999562553Hz 1.000ns
ELCR: 0EA0
pit0: hz 1193182 max 50000000000000 min 100000000000 mul 5120 58207
Enable asynchronous printing
Acedia       1.374s  i8250attr 20
uart         1.446s  Trying i8250uart COM1
uart         1.536s  Trying i8250uart COM2
uaUarttask   1.628s  uartmsg get attr *
rUarttask    1.714s  uartmsg get attr b
Uarttask     1.798s  i8250attr 2
tuart        1.869s  i8250attr 8
Uarttask     1.938s  uartmsg set attr d
Uarttask     2.020s  i8250attr 8
             1.445s  Starting
uart         1.446s  Allocate memory
uart         1.446s  uartdiscover
uart         1.625s  uartdiscover: uarts 0xf023d5a0, uartdir 0xf023d5e8
uart         1.626s  uartdiscover: 2 entries
uart         1.627s  uartopen /dev/uart/eia0: COM1
rread attr=*: 65 bytes: '# * b c d e f h i k l m p r s x framing overruns berr cts dsr dcd'
rread attr=b: 4 bytes: '9600'
10 seconds type ahead:
huart        12.974s  i8250attr 8
rrUarttask   13.043s  uartmsg set attr d
Uarttask     13.127s  i8250attr 8
ead: 1 bytes: 'h'

This was Osprey
```