

# Solaris 10 Performance Features

**Tom Gendron**  
**Technical Specialist**  
Sun Microsystems

# Agenda

- Introduction
- Networking Performance
- Other Performance Features

# Introduction

- Loss of market share to Linux
  - > Slower hardware performance
  - > Slower Solaris (software) performance
- Key software workloads
  - > Web
  - > SSL
  - > Java
  - > System call performance

# Introduction

- Traditional Solaris focus:  
Large machines
- New focus:
  - > Small machines
  - > Opteron based servers and workstations
- Still interested in scale up

# Networking Performance

# Prior Solaris Networking

## How slow was it

- Solaris 3x slower than Linux  
SPECweb99
- Some system calls 10x slower than Linux  
Libmicro
- Single CPU could not drive 10+Gb/sec.
- Various benchmarks showed 40-50% behind

# Prior Solaris Networking

## Why so slow

- Three Key areas:
  - 1.) *STREAMS* overhead
  - 2.) IP Exclusive lock
  - 3.) Connection setup/tear down

# Solaris 10 Networking

## Project FireEngine

- Improve networking performance
- Out of Box performance
- NCA to use core TCP/IP stack
- Stack sustainability and observability
- Feature growth management

# Solaris 10 Networking

- Results:
  - > 45% improvement SPECWeb99
  - > 40% improvement NetPerf
  - > 17%-58% improvement in setup/tear down
    - > bind(), close(), connect(), listen(), socket()
- Efficiency
  - > Drive a 1-Gb NIC with < 10% utilization (V20z)
  - > Drive a 10-Gb Nic with ~ 45% utilization (V20z)
- Beat RH 3 by 30% with Apache

# Solaris 10 Networking

## Project FireEngine

- Merged TCP/IP
- Vertical Perimeters
- IP Multi-Threading
- IP Classifier
- Connection setup and tear down

# Solaris 10 Networking

## Project FireEngine

- Merged TCP/IP
  - > TCP/IP -> 1 *STREAM* module
  - > Use a function call interface
  - > Better data and instruction locality

# Solaris 10 Networking

## *Merged TCP and IP*

- Separate *STREAMS* modules
  - > Overhead from putnext() call
  - > Call stack depth
  - > Context switching
  - > Repeated structure initialization (mblocks)
  - > Poor code/data locality

# Solaris 10 Networking

## Project FireEngine

- Vertical Perimeters (queue)
  - > A common serialization queue for In/Out packets
  - > Provides lockless mutex for all TCP connections
  - > PtoP from Socket() to NIC (In/outbound packets)

# Solaris 10 Networking

## Project FireEngine

- Vertical Perimeters and Data Locality
  - > Bind each inbound connection to a queue
  - > Bind inbound queue thread to interrupted CPU
  - > Bind each outbound connection to a queue
  - > Bind outbound queue thread to CPU application is on

# Solaris 10 Networking

## Project FireEngine

- IP Multithread
  - > *STREAMS* IP goes coarse exclusive & often
  - > FE uses standard MT locks and queue
  - > Now IP fine-grained exclusive

# Prior Solaris Networking

## IP MT

- IP goes exclusive on:
  - > qwriter()
  - > During plumbing
  - > most xxx\_set\_ioctl(), other operations too.
  - > Can take long to return to MT mode
- Assumption that this is infrequent is wrong

# Solaris 10 Networking

## Project FireEngine

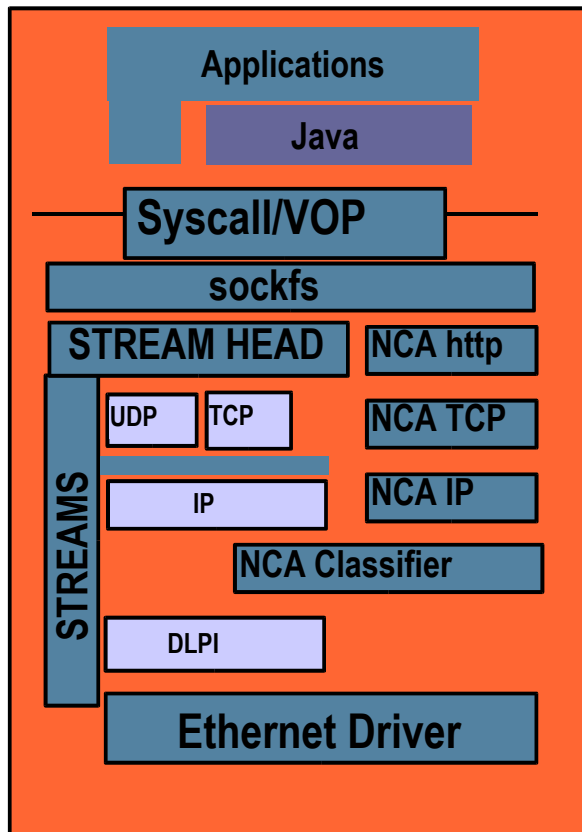
- IP Classifier based Fan-out
  - > Class of packet determines:
  - > Connection structure
  - > Instance of queue
- New incoming connections
  - > Attach to interrupted CPU
  - > Fan out across all CPUs
  - > Both interrupt mode and polling supported

# Prior Solaris Networking

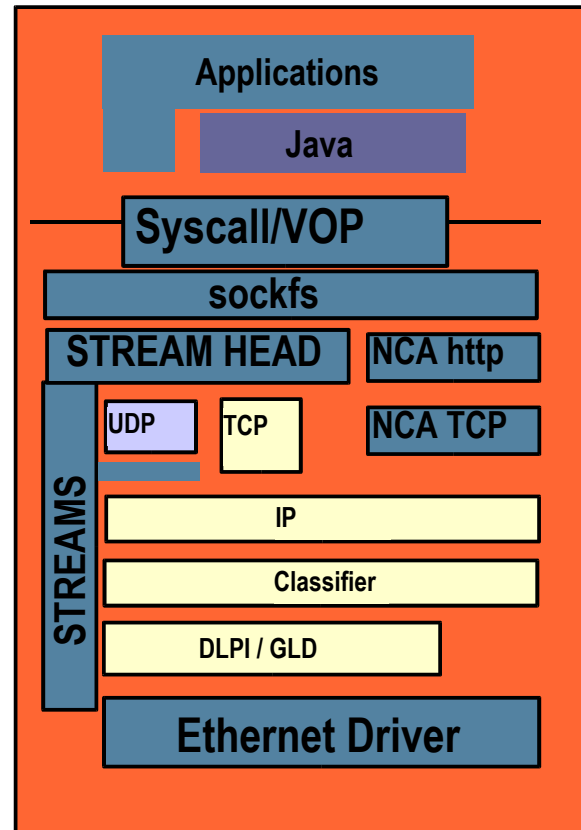
## TCP connection setup/tear down

- Socket related system calls
  - > Overhead due to plumbing a STREAM
  - > Mutex on Listener during plumbing
  - > Redundent structure allocation/initialization
  - > Multi perimeter packet processing
- Many improvements due to merged TCP/IP
- Phase II will show more

# Network Stack Evolution



**Solaris 9**



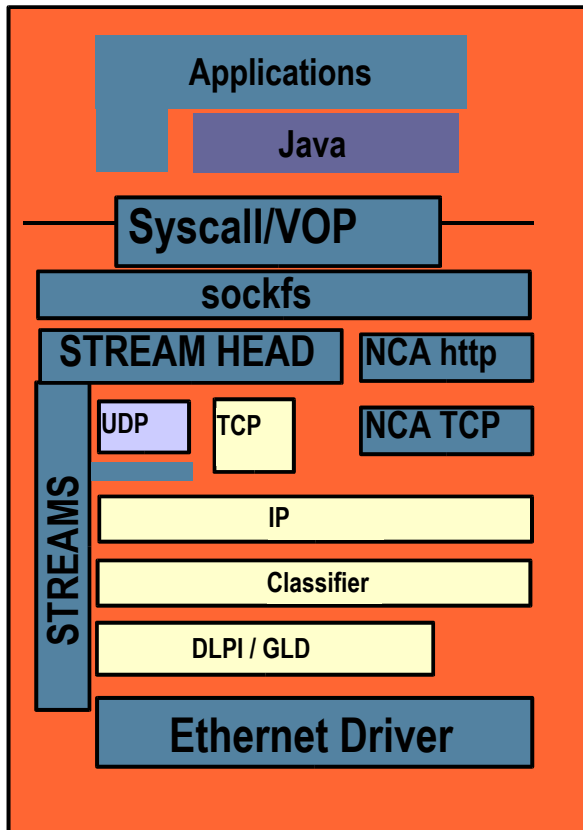
**Solaris 10  
Phase I**

# Solaris 10 Networking

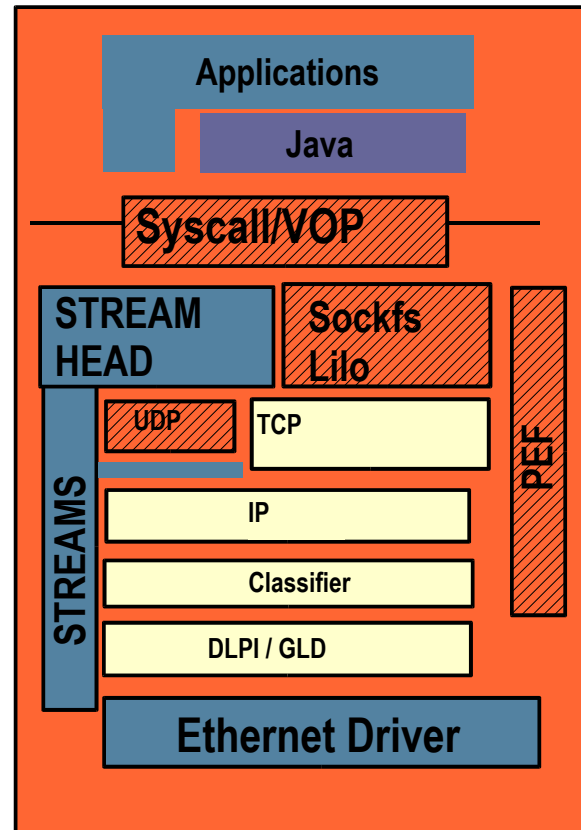
- Project Phases:

	Phase 1	Phase 2	Phase 3 (Firehose)
Release	Solaris 10	Solaris 10 updates and Solaris Express	Solaris 11
Feature	IP MT IP Fan-out classifier Vertical perimeters Merged TCP/IP Syscall changes -Accept -Connect -Socket -Sendfilev -close	Full NCA merge UDP Performance Event list Packet event framework Classifier offload  TOE/RDMA Support Observability	TLI/TPI Full classifier Other offload

# Network Stack Evolution



**Solaris 10  
Phase I**



**Solaris 10  
Phase II**

# Performance Projects

- Dynamic (per process) TSBs
  - > Run 20k+ procs
- Kernel Cryptographic Framework
- Fast fsync/fsflush
  - > Great for large systems
- InfiniBand support
- AMD64 support

# Performance Projects

- Fast TCP loopback
- Fast & scalable Unix domain sockets
- NFS V3 server and client performance improvements
- Faster boot
- Unified process model
- 10Gb drivers

# New APIs

- Event Ports
  - > New event completion mechanism unifies poll, async IO, timers and messaging to provide scalable, thread-safe event dispatching
  - > Handling 20,000 connections is now cheap
  - > See `port_create` man page for details
- libc atomic data manipulation functions

# New APIs

- New Igroup (MPO) functions in liblgrp
  - > See liblgrp man pages for details
  - > Can be a good win for memory bound apps on high-end SPARC systems and Opteron
- Libumem
  - > Scalable fast memory allocator for multi-threaded applications
  - > 10x performance improvements seen for some Wall Street C++ apps
  - > If your app is multi-threaded, you *need* to link with a threaded malloc

# More on libumem

- Application memory leaks? Heap corruption?
- `libumem` is more than a scalable memory allocator:

```
# export UMEM_DEBUG=default
# export UMEM_LOGGING=transaction
# export LD_PRELOAD=libumem.so
# ./buggy
Abort(coredump)
# mdb buggy core
Loading modules: [ libumem.so.1 libc.so.1 ld.so.1 ]
> $c
libc.so.1`_lwp_kill+0x15(1, 6)
libc.so.1`raise+0x1f(6)
libumem.so.1`umem_do_abort+0x25(0, d13b5000, 8046924, d1397b8a, d13a2f3c,
d13b9a00)
libumem.so.1`umem_err_recoverable+0x46(d13a2f3c, d13b9a00, 808fe88, d13a2f64)
libumem.so.1`process_free+0x82(808fe88, 1, 0, 804696c, 805096f, 808fe88)
libumem.so.1`free+0x14(808fe88, 8046964, d13de044, d135fc95)
main+0x77(1, 8046990, 8046998)
_start+0x80(1, 8046b74, 0, 8046b7a, 8046b82, 8046ca5)
```

# Libumem cont...

```
> ::findleaks -v
findleaks:                maximum buffers => 117
findleaks:                actual buffers => 104
findleaks:
findleaks:                potential pointers => 20687
findleaks:                dismissals => 16204           (78.3%)
findleaks:                misses => 4176             (20.1%)
findleaks:                dups => 304                ( 1.4%)
findleaks:                follows => 3                ( 0.0%)
findleaks:
findleaks:                elapsed wall time => 0 seconds
findleaks:
```

CACHE	LEAKED	BUFCTL	CALLER
08079810	1	0808ed50	foo+0x25
0807cc10	1	08090dd0	foo+0x25
...			
08084010	14	080ad9b0	foo+0x25
08084210	41	080ad320	foo+0x25
08084410	18	08156ee8	foo+0x25
0807cc10	1	08090e48	main+0x48

-----

Total 101 buffers, 654152 bytes

```
> 080ad9b0::bufctl_audit
```

ADDR	BUFADDR	TIMESTAMP	THREAD
	CACHE	LASTLOG	CONTENTS
80ad9b0	80b3000	e84877b5766	1
	8084010	806ba8c	0
	libumem.so.1`umem_cache_alloc_debug+0x16c		
	libumem.so.1`umem_cache_alloc+0x15c		
	libumem.so.1`umem_alloc+0x3f		
	libumem.so.1`malloc+0x23		
	foo+0x25		
	main+0x34		
	start+0x80		

# Platform and CPU optimizations

- SMT support (Pentium-4 and Niagara)
  - > Halt idle threads, spin loop pause statements, load balance across chips
  - > Added virtual processor concept to *psrinfo -vp*
  - > 2x Pentium-4 power savings: 45%
- CMP support (UltraSPARC-IV+, x64)
  - > Scheduler is CMP aware
  - > Load balanced across chips
- MPO support for UltraSPARC-IIIi and Opteron in addition to 6800 and F15Ks
  - > Big win for memory intensive apps

# Platform and CPU optimizations

- Faster string functions for all platforms
- x86 specific: SSE2 support, MMX and SSE2 tuned copy & bzero
- x64 has faster system calls, processor-specific libc
- New compilers & options used for SPARC, x64 & x86
- More flexible CPU-specific libraries support

# Some Early Performance Results

- Networking
  - > 30-50% improvement web serving, ttcp
  - > Saturates 1 Gb Ethernet with 8% of Opteron
- NFS
  - > 60% improvement for NFS V3
- SunRay
  - > Better scalability, faster response (testing still underway)
- x64
  - > 64 bit applications are typically 10-15% faster

# Solaris 10 Performance Features

**Tom Gendron**

tom.gendron@sun.com