

SOLARIS

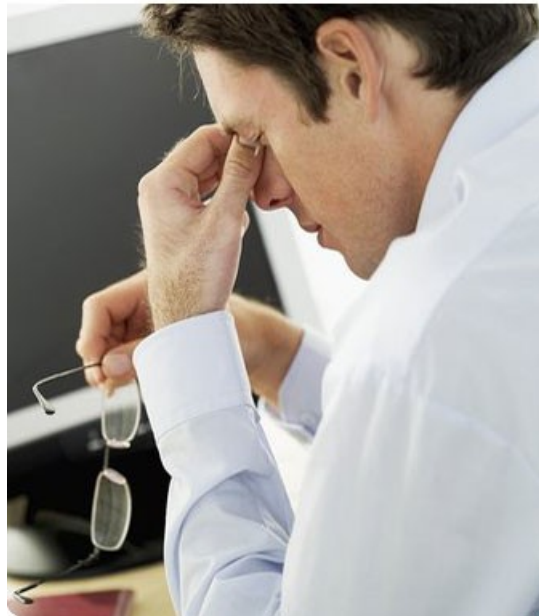
Carlos Piedrafita
Solutions Architect
Sun Microsystems, Inc.

Sun's Strategy

Grow Communities,
Increase Participation
and Opportunity



Address Our
Customers' Biggest
Pain Points



Innovate the World's
Best Infrastructure
Technology



Communities Create Markets

What's Behind the Strategy



Developers

- Build a better open platform on which developers can create value and innovate
- Provide free and easy access
- Utilize the power of community
- Intellectual property protection and indemnification
- Be the preferred platform for web creation partners & for business integration



Infrastructure Buyers

- Innovate: Build a better network services delivery platform
- No lock-in: Multi OS, Multiplatform everything
- More scalable, secure and open
- Best of industry standard coupled with innovation
- Extend our reach with a family of partners



Citizenship

- Investing in the community
- Sharing opportunity and educating
- Eliminating the digital divide
- Being eco-responsible

Strategic Battleground

Network
Services



Operating
Systems



Architecture



About Three Years Ago...

Operating
System



Architecture



Today's Reality

Superior Choice, Value, Innovation

Operating
System



solaris opensolaris™

Architecture



ULTRASPARC™

Solaris for 500+ Platforms = Choice



Solaris Investment Protection

**Guaranteed Source
Compatibility**

**SPARC to x86/AMD64
x86/AMD64 to SPARC**

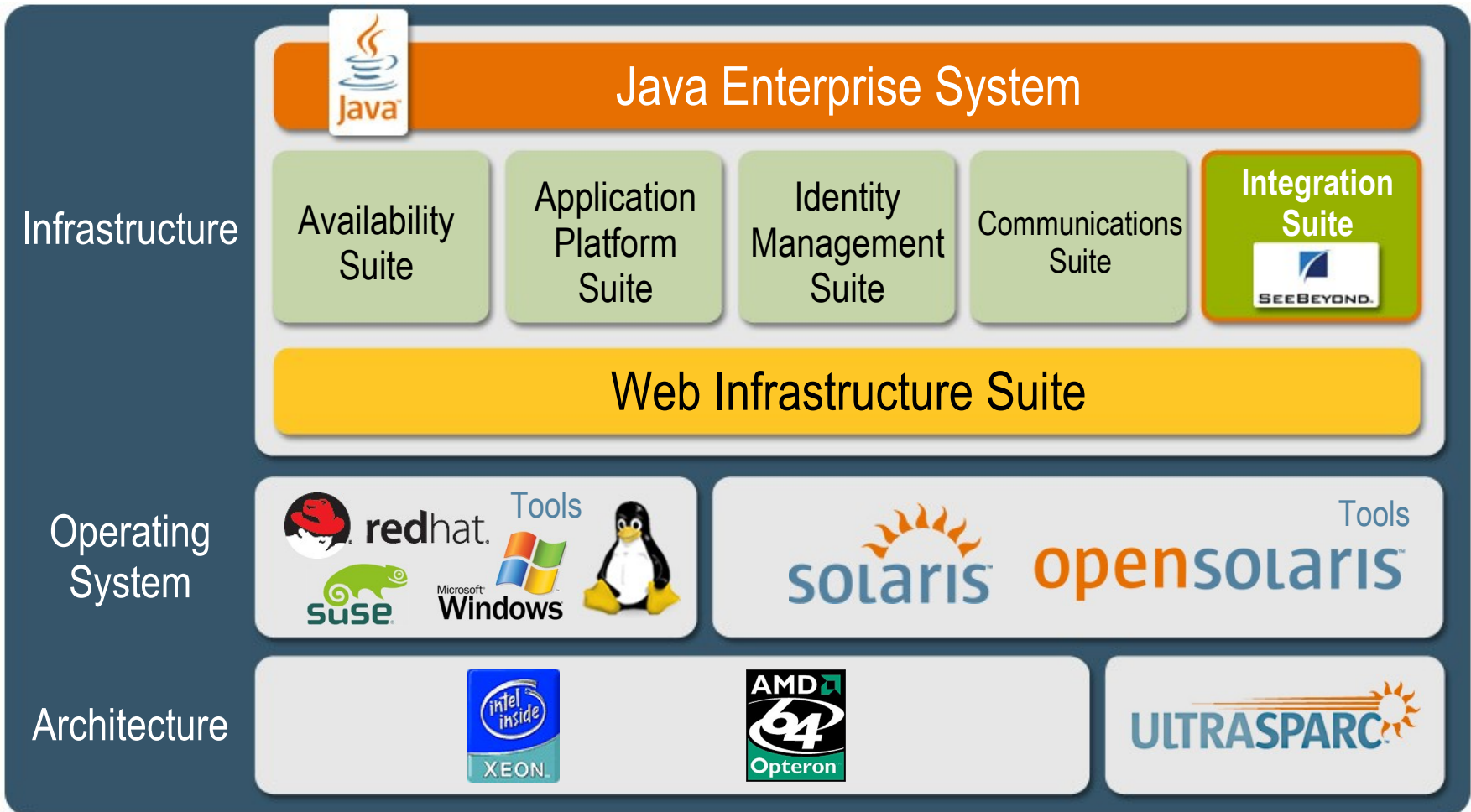
Solaris Investment Protection

**Guaranteed Binary
Compatibility
for 7 Years**

Solaris 2.6 to Solaris 10

Java Enterprise System

A Complete Network Services Platform



Solaris Enterprise System

Solaris 10

Java Developer Tools
Studio Tools



Java Enterprise System
Web Server
App Server
Directory Server
Identity Management
Portal Server
Messaging Server
Clustering

Open Source Database

N1 Systems and
Services Management

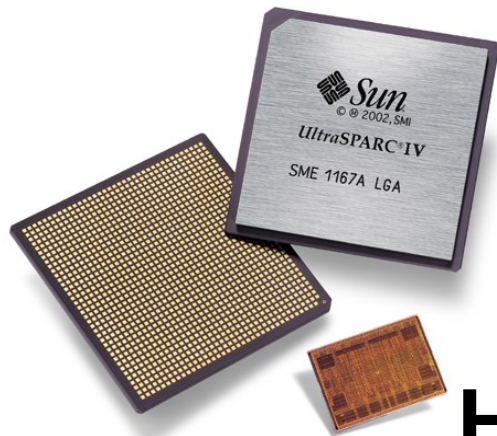
The Solaris logo, featuring a stylized orange sun with rays above the word "SOLARIS" in blue, with "Free and Open" in orange below it. The word "SOLARIS" has a trademark symbol (TM) and is reflected below it.

SOLARIS™
Free and Open

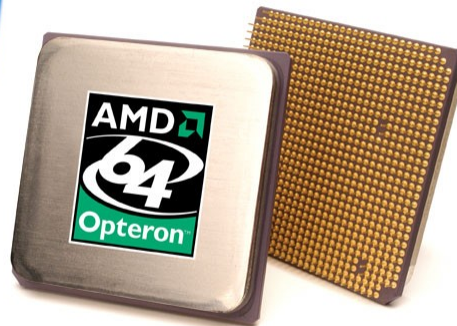
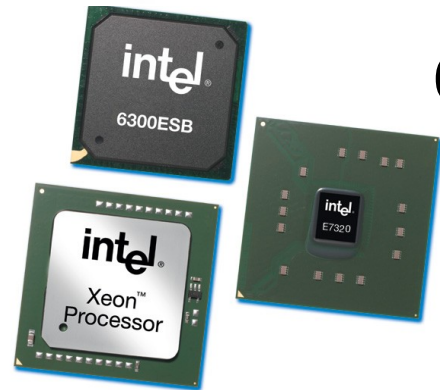
A Multi-Platform OS Strategy



A Multi-Platform OS Strategy

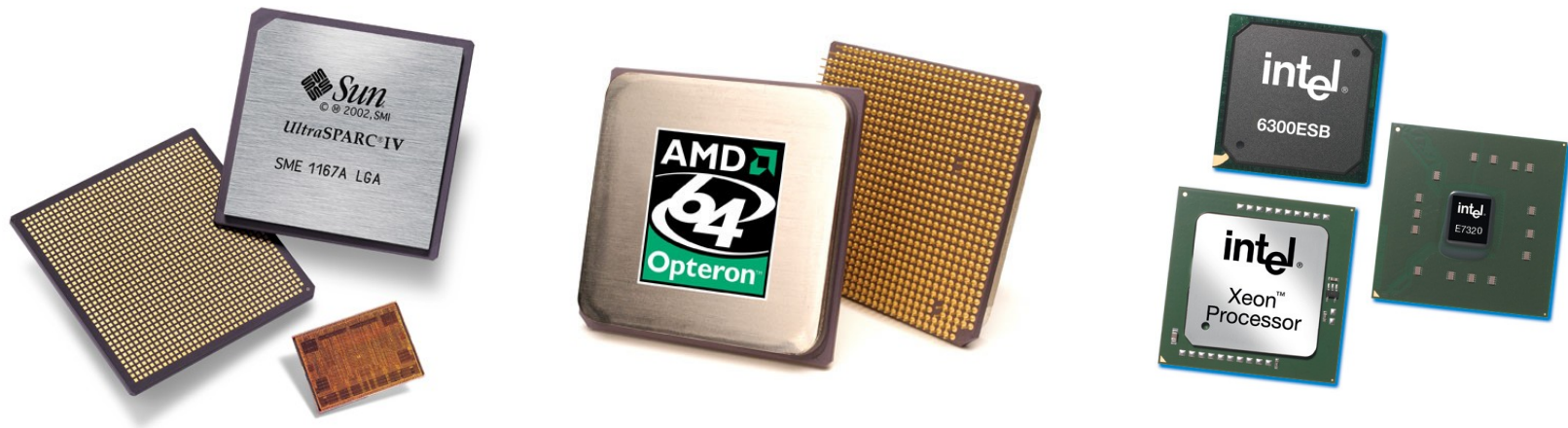


**Hundreds
of Systems**



**Thousands of
ISVs and Partners**





Hundreds of Systems



Thousands of ISVs and Partners

ISV Community Delivers on Solaris 10



Computer Associates®



IBM. DB2 Information Management Software

IBM. WebSphere. software



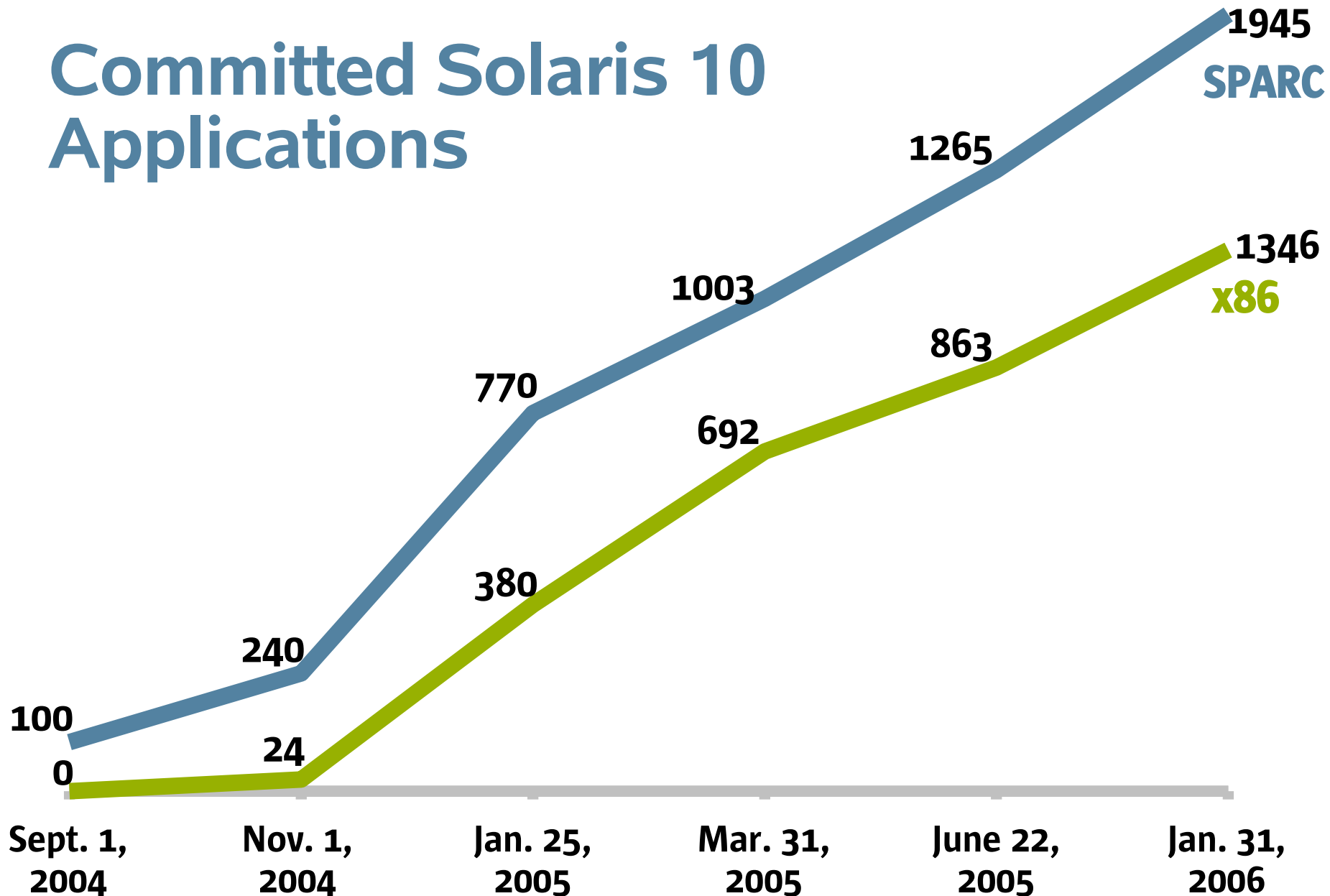
IBM. Rational. software



IBM. Tivoli. software



Committed Solaris 10 Applications



Solaris 10 License Growth

4M
Solaris 10
Licenses



ULTRASPARC

x64, x86

IBM



DELL



3/05

1/06

Solaris 10: A Generation Ahead



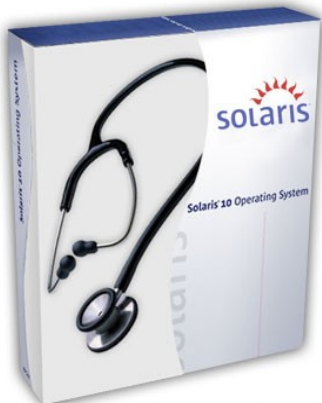
Extreme Performance



Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

Solaris 10: A Generation Ahead



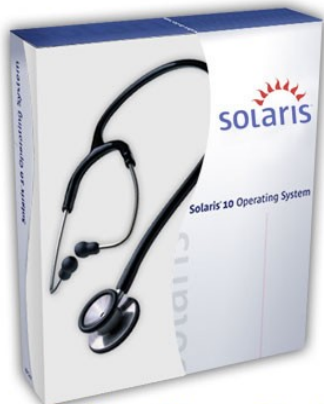
Extreme Performance



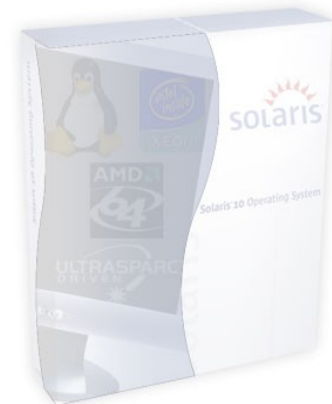
Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

Predictive Self Healing

- Solaris Fault Manager
- Solaris Service Manager
- Automatic fault diagnosis, isolation and recovery
- Reduces
 - > Hardware failures
 - > Application failure impact
 - > Application Downtime



Solaris Fault Manager

- Automated error handling
 - > Detect faults
 - > Aggregate faults
 - > Diagnose faults
 - > Report faults
 - > Mitigate faults



Solaris Fault Manager

- Reduces downtime
 - > Components offlined before failure
- Reduces complexity
 - > Simplified error reporting
- Reduces costs
 - > 24 x forever uptime
 - > Increased utilization
 - > Higher server-to-administrator ratio



Relentless Availability

Solaris Fault Manager

Traditional Error Handling

- Each component operates independently to handle and communicate errors
- Humans try to diagnose fault, impact, and appropriate corrective action

WARNING: /io-unit@fe0200000/sbi@0,0/dma@0,81000/esp@0,8000000 (esp0):
Connected command timeout for Target 0.0

NOTICE: correctable error detected by pci0 (upa mid 1f) during DVMA
read transaction AFSR=40f40000.1f800000 AFAR=00000000.a25b4000 ...

[AFT0] errID 0x0000004d.23105c04 Corrected Memory Error on U1004 is
Intermittent
[AFT0] errID 0x0000004d.23105c04 ECC Data Bit 14 was in error and
corrected



Solaris Fault Manager

Fault Management Architecture

- Include impact and action statements in a consistent format
- All events are managed and coordinated through a single management service

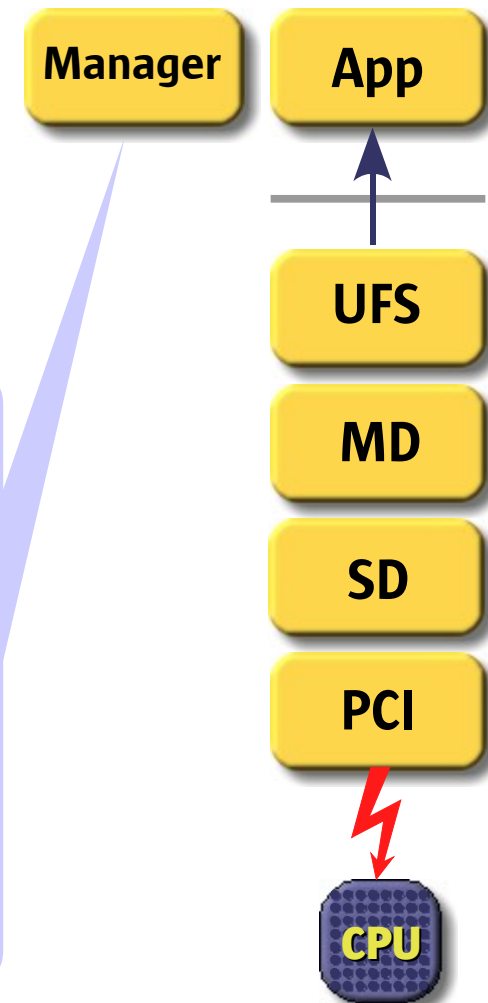
SUNW-MSG-ID: SF20000-W84N-KP3A-TF; TYPE: Fault,
VER: 1, SEVERITY: Minor

AUTO-RESPONSE: Removal of the faulty memory resources
has been initiated

IMPACT: Reduction in available memory resources

REQ-ACTION: A service call should be scheduled to
inspect/replace the suspect components

DESC: A correctable memory data error occurred which has
been diagnosed to be caused by a fault in a memory hardware
component.



Solaris Fault Manager

Message code

- Customer web-site will provide latest repair procedures for each diagnosis
- Links to information on latest FMA capabilities, updates and plans

sun.com/msg/SF20000-W84N-KP3A-TF

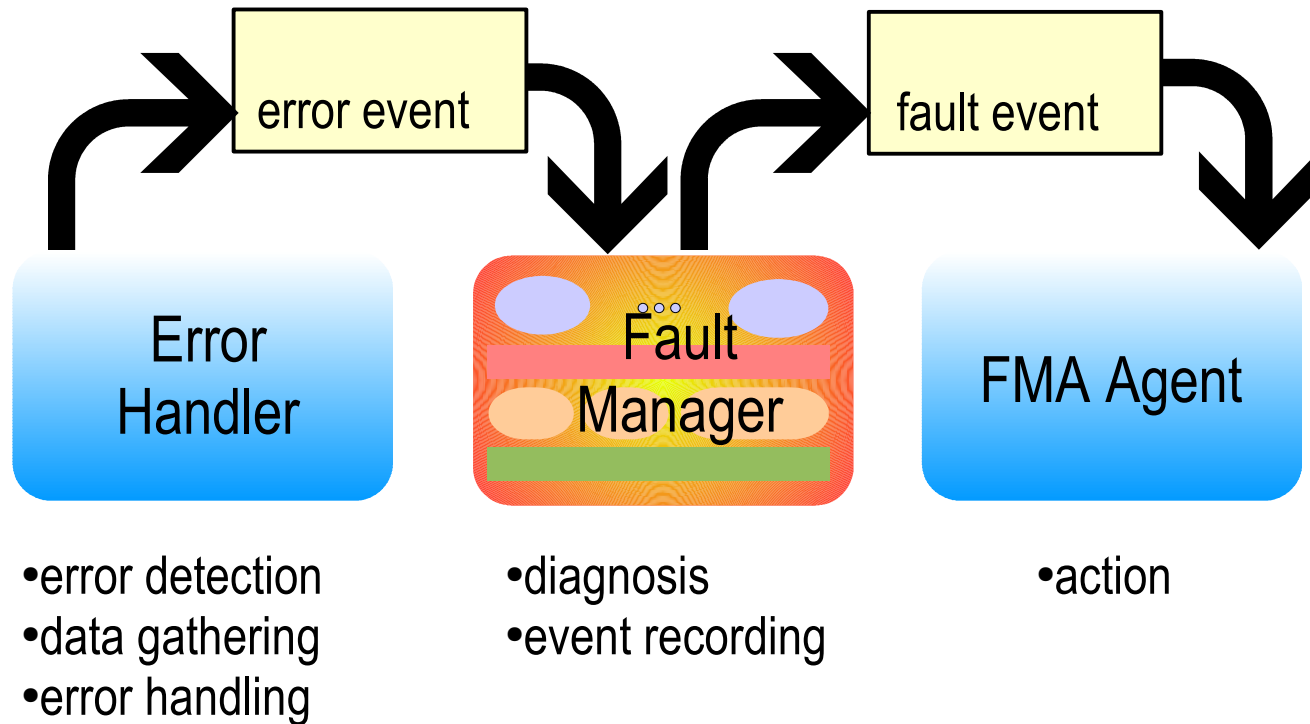
SUNW-MSG-ID: SF20000-W84N-KP3A-TF; TYPE: Fault,
VER: 1, SEVERITY: Minor

AUTO-RESPONSE: Removal of the faulty memory
resources has been initiated



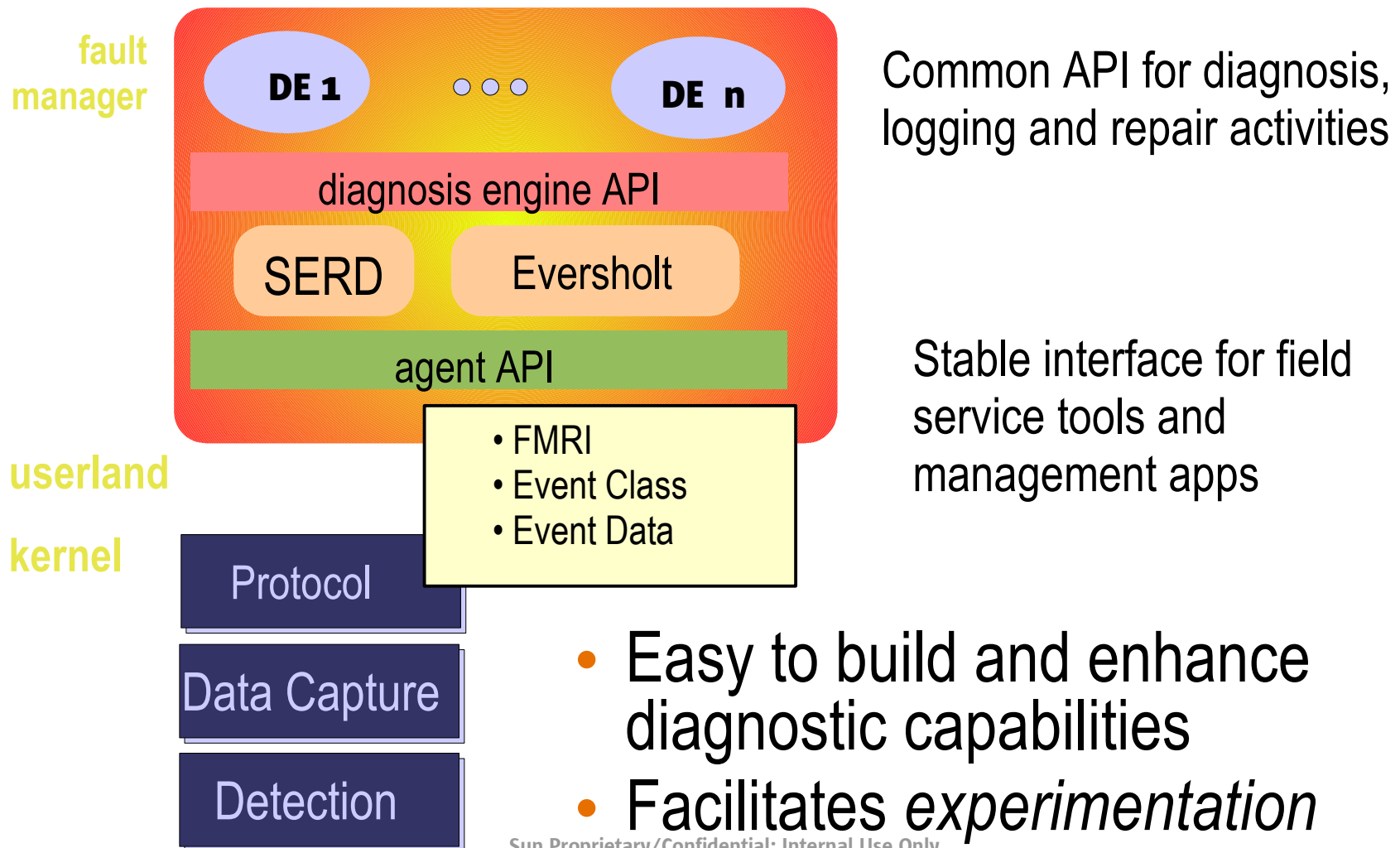
Solaris Fault Manager

Fault Management Flow



Solaris Fault Manager

Solaris Deployment



Solaris Service Manager

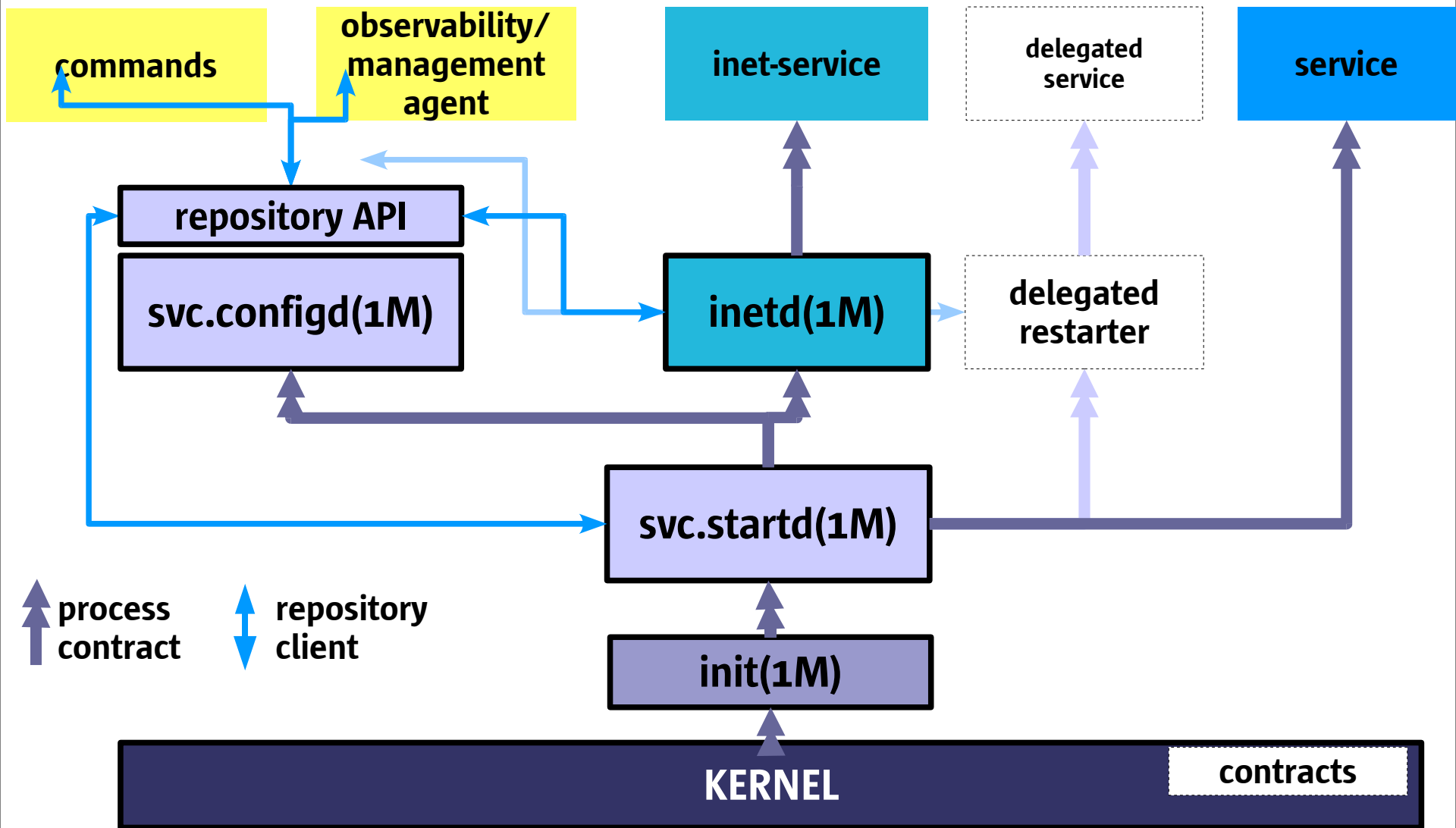
- Defines relationships among:
 - > Applications
 - > Solaris components
- Simplifies administration
 - > Consolidates “application profile”
 - > Helps manage components
- Eases installation
- Increases service reliability
 - > Detects service outages quickly
 - > Recovers services accurately



Relentless Availability

Solaris Service Manager

Architecture



Solaris Service Manager

Existing

/dev/*
network interfaces

low level devices

/etc/inittab
/etc/init.d/*
/etc/rc?.d/*
/etc/inet/inetd.conf

invocation,
termination

/etc/system
/etc/default/*
/etc/inet/*
/etc/hostname*
/etc/dhcp/*
/etc/ppp/*

properties

file system services

security

Name Server backends
local files

repository

Solaris Management Framework

device services

service methods
service dependencies
milestones

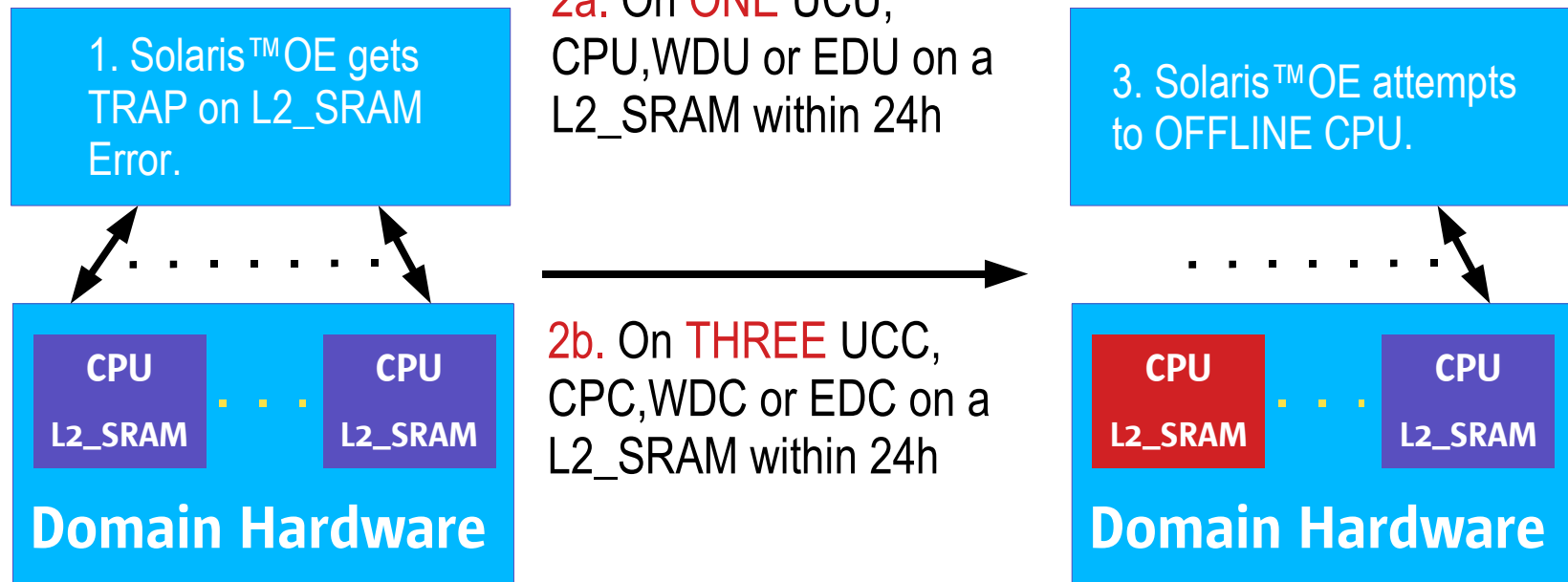
service properties
/var/svc
/lib/svc

entity authorizations
delegated roles
security profiles

remote datastores
local cache

Automatic fault diagnosis

Automatic CPU Off lining



Domain Console (successful):

Feb 3 6:38:40 A SUNW,UltraSPARC-III: NOTICE:[AFT1]CPU5
offlined due to more than 2 xxC Events in 24:00:00(hh:mm:ss)

Domain Console (unsuccessful):

Feb 3 6:46:07 A SUNW,UltraSPARC-III: NOTICE:[AFT1]**Failed to**
offline CPU7 due to more than 2 xxC Events in 24:00:00
 (hh:mm:ss),will try again

Solaris 10: A Generation Ahead



Extreme Performance



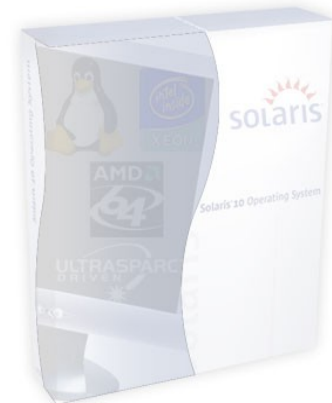
Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

Performance

% Improvement, Solaris 9 vs. Solaris 8



40%



**LDAP
Overall**

55%



NFS

87%



UFS

100%



MPSS

200%



JVM

200%



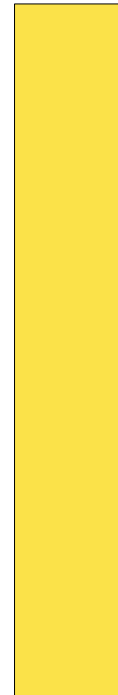
**Mozilla/
SunRay**

400%



**LDAP
Search**

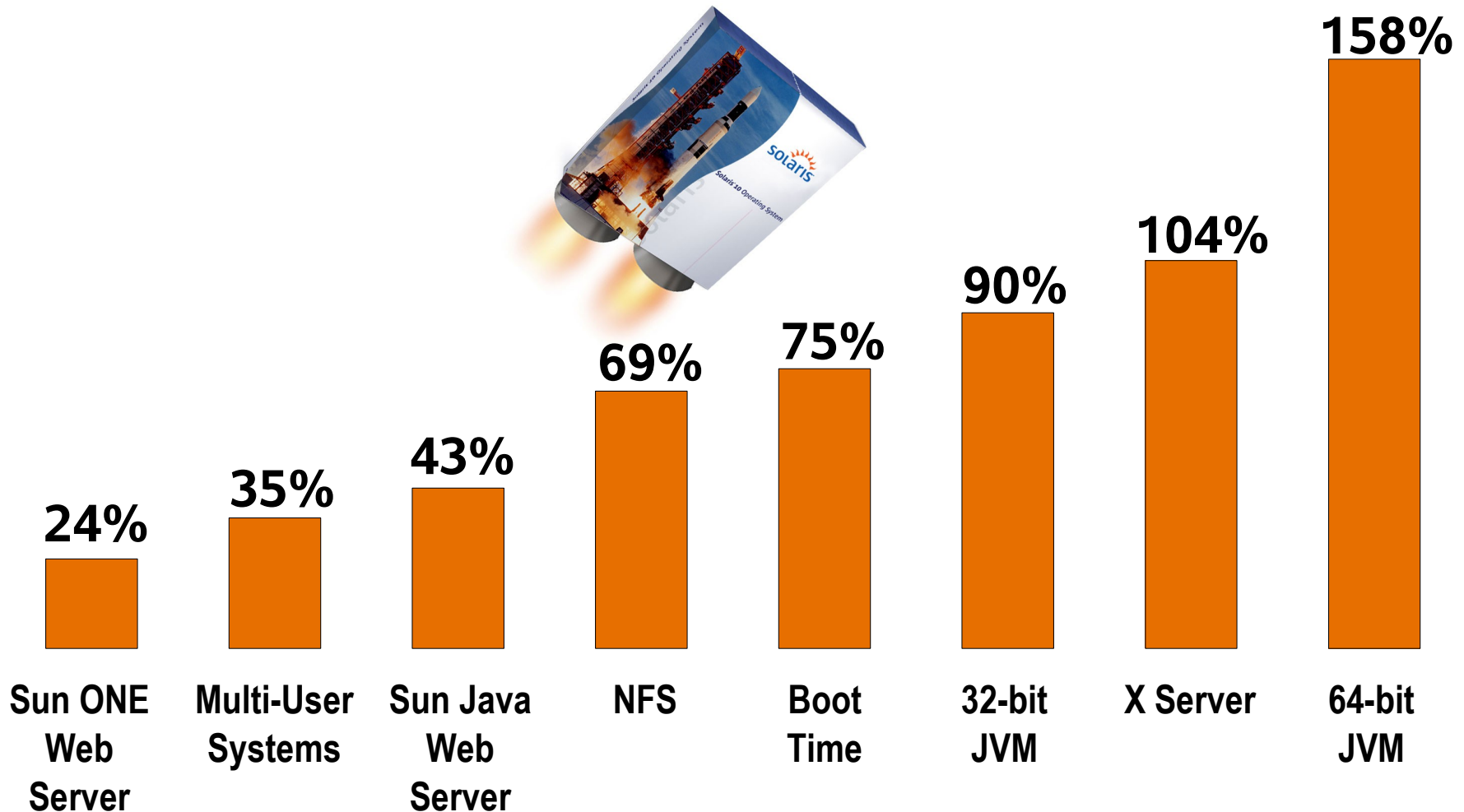
400%



Threads

Performance

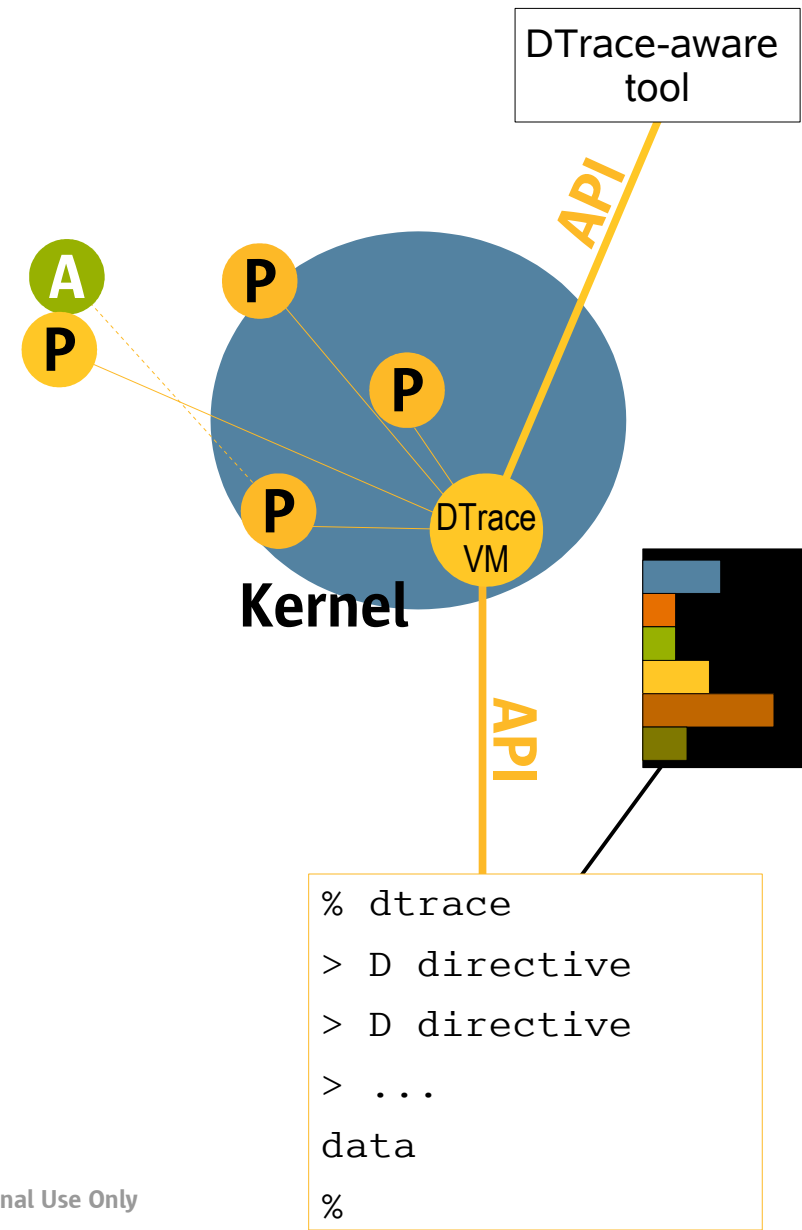
% Improvement, Solaris 10 vs. Solaris 9



Dynamic Tracing

Designed for Production Systems

- Safe; always there
 - > No performance hit
 - > No app or OS changes
- Views system as a whole
 - > Comprehensive
 - > Extensible; scriptable
- Debug, analyze, optimize in real time
 - > Solutions in minutes or hours, not days or weeks
 - > 3-30x speedups seen



DTrace Can Improve Performance

Simple Tool, Extreme Performance

Maximum performance, minimal time:

+32% Financial Database (Before lunch)

+35% Online Parcel Tracking System (In four hours)

+80% Futures Forecasting Application (In an afternoon)

+267%

Message Handling Benchmark
(In two days)

+300%

Data Routing Application
(In 5 hours)



Dynamic Tracing

Dynamic Instrumentation

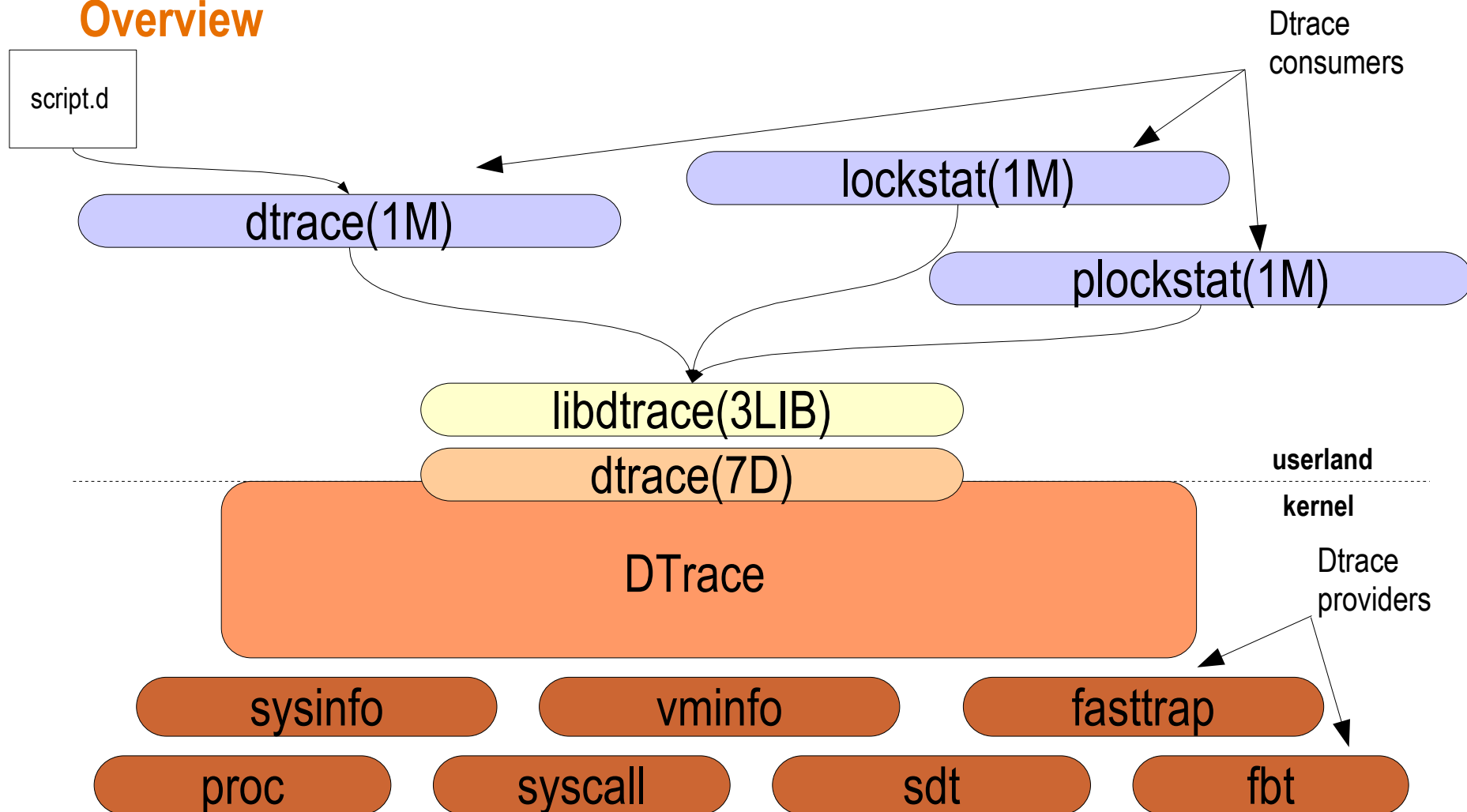
- Want to be able to *dynamically* modify the system to record *arbitrary* data
- Must be able to do this on a *production* system
- Must be completely *safe* – there should be no way to induce fatal failure

The DTrace Revolution

- Tightens the diagnosis loop:
hypothesis → *instrumentation* → *data gathering*
→ *analysis* → *hypothesis*
- Tightened loop effects a revolution in the way we diagnose transient failure
- Focus can shift from *instrumentation* stage to *hypothesis* stage:
 - > Much *less* labor intensive, less error prone
 - > Much *more* brain intensive
 - > *Much* more effective! (And a *lot* more fun)

Dynamic Tracing

Overview



Dynamic Tracing

Probes

- A *probe* is a point of instrumentation
- A probe:
 - > Is made available by a *provider*
 - > Identifies the *module* and *function* that it instruments
 - > Has a *name*
 - > Is assigned an integer identifier
- A probe is uniquely identified by its *provider:module:function:name*

Dynamic Tracing

Providers

- A provider represents a methodology for instrumenting the system
- Providers make probes available to the DTrace framework
- DTrace informs providers when a probe is to be enabled
- Providers transfer control to DTrace when an enabled probe is hit

Dynamic Tracing

Providers (2)

- Dtrace has quite a few providers, eg.:
 - > The *function boundary tracing (FBT)* provider can dynamically instrument every function entry and return in the kernel
 - > The *syscall* provider can dynamically instrument the system call table
 - > The *lockstat* provider can dynamically instrument the kernel synchronization primitives
 - > The *profile* provider can add a configurable- rate profile interrupt of to the system

Dynamic Tracing

Providers (3)

- > The *vm*info provider can dynamically instrument *the kernel* “*vm*” statistics, used by commands such as *vmstat*
- > The *sys*info provider can dynamically instrument the kernel “*sys*” statistics, used by commands such as *mpstat*
- > The *pid* provider can dynamically instrument application code, such as any function entry and return point (actually any instruction!)
- > The *io* provider can dynamically instrument disk I/O events

Dynamic Tracing

Consumers

- A DTrace consumer is a process that interacts with DTrace
- No limit on concurrent consumers; DTrace handles the multiplexing
- `dtrace(1M)` is a DTrace consumer that acts as a generic front-end to the DTrace facility
- `lockstat(1M)` and `plockstat(1M)` are consumers

Dynamic Tracing

The D language

- D is a C-like language specific to DTrace, with some constructs similar to awk(1)
- Complete access to kernel C types
- Complete access to statics and globals
- Complete support for ANSI-C operators
- Support for strings as first-class citizen
- We'll introduce D features as we need them...

Dynamic Tracing

The D language

- For example, a script to trace the executable name upon entry of each system call:

```
#!/usr/sbin/dtrace -s

syscall:::entry
{
    trace(execname);
}
```

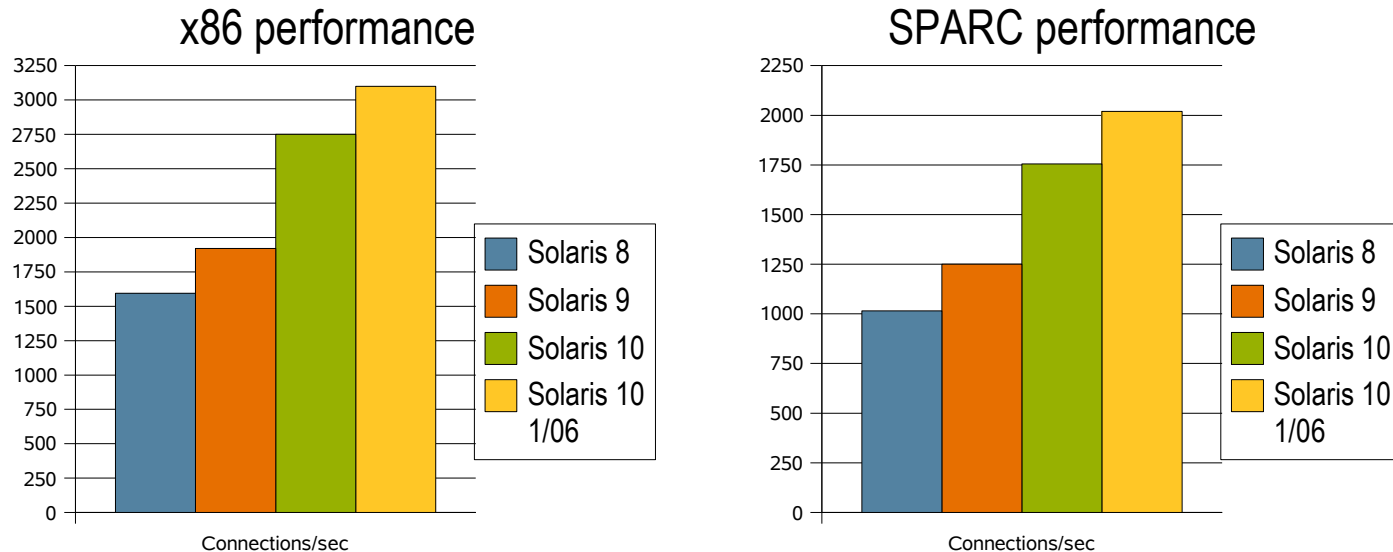
IP Stack Streamlining

- Goal: saturate 10 Gb Ethernet with reduced overhead
- Classification engine for IP packets
- Specific, efficient methods to handle different packet types
- Solaris Network Cache and Accelerator functionality merged into IP stack



Extreme Performance

Solaris Network Performance



- Performance has doubled since Solaris 8
 - > Improvements in Solaris 9, Solaris 10, and Solaris 10 1/06
- Now on par with Linux performance and still improving

Generic Lan Driver V3

- Nemo Dynamic Interrupt vs. Polling
 - > Bind a NIC to a Squeue and let the Squeue own the NIC and have the ability to turn off interrupts
 - > If Squeue becomes backlogged, it turns the NIC interrupts off
 - > If no backlog, Squeue switches the NIC back to interrupt mode
- 10 Gbit Ethernet
 - > We can do 7.3 Gbps on a v20z (with 50% utilization) using 9k frames
 - > Solaris 10 set new LAN record during *Internet 2: Land speed record* challenge by transferring 14Gbps over 2 x 10Gbps using a v20z
- Trunking
 - > Each member of the trunk is owned by individual Squeues which control the rate of arrival of packets
 - > We see pretty linear scalability for a trunk of 4 1Gb NICs – 3.6Gbps

Scale Up, Scale Out

- Traditional focus: scalability
- Not always optimal for latency
- Goal: Industry-leading Solaris performance on all systems, including 1P/2P



Extreme Performance

Scaling Up

Double Your Systems's Performance Over Lunch



- UltraSPARC-IV: Solaris CMT optimization yields 2x system improvement
- Web on midframe systems: 47% gain
- Multi-user systems: 42% gain
- Scientific apps: 37% gain
- System boot time: 65% faster
- 100,000+ active apps per server
- Higher service levels through Dynamic System Domains + Solaris Containers

Scaling Out

- Extreme Performance
 - > Wire speed for 10 Gb Ethernet
 - > 20-52% increase for Java App Servers
 - > 30% increase for Web Servers
- Choice of Platforms
 - > One Solaris
 - > Standard, volume platforms
 - > SPARC & x86-x64
- Seamless Scaling
 - > New 64 bit support for AMD64
 - > Solaris containers on any system



Throughput Computing Ready

- Ready for *multi-threaded* applications
- Ready for *multiple* applications
- MT-hot kernel, Java VM, system libraries
- Fast thread library
- 64 bit memory addressing
- Memory Placement Optimization
- Development & tuning tools



Extreme Performance

Solaris 10: A Generation Ahead



Extreme Performance



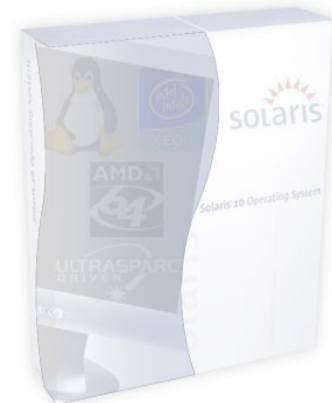
Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

Solaris ZFS

- Streamlined system administration
 - > Efficient resource allocation via storage pools
 - > Automates administrative tasks
 - > Extensible: add features such as encryption
- Self-healing data
- Virtually unlimited capacity
 - > 16 *billion billion* times greater than today
- Breakthrough performance



Optimal Utilization

Solaris ZFS

Design principles

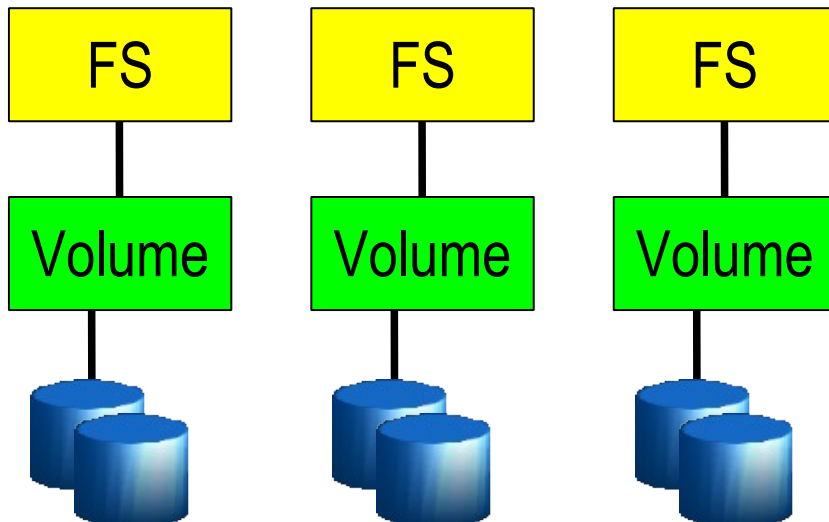
- Pooled storage
 - > Does for storage what VM did for memory
- End-to-end data integrity
 - > Historically considered “too expensive”
- Everything is transactional
 - > Keeps things always consistent on disk
 - > Removes almost all constraints on I/O order

Solaris ZFS

FS/Volume Model vs. ZFS

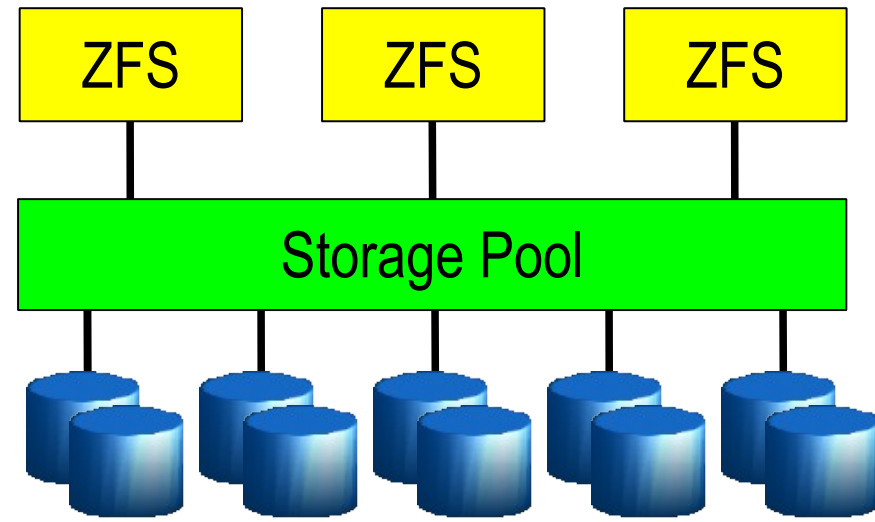
Traditional Volumes

- Partition/volume for each FS
- Grow/shrink by hand
- Each FS has limited bandwidth
- Volumes cause stranded storage



ZFS Pooled Storage

- No partitions to manage
- Grow/shrink automatically
- All bandwidth always available
- Pool allows space to be shared



Solaris ZFS

FS/Volume Model vs. ZFS

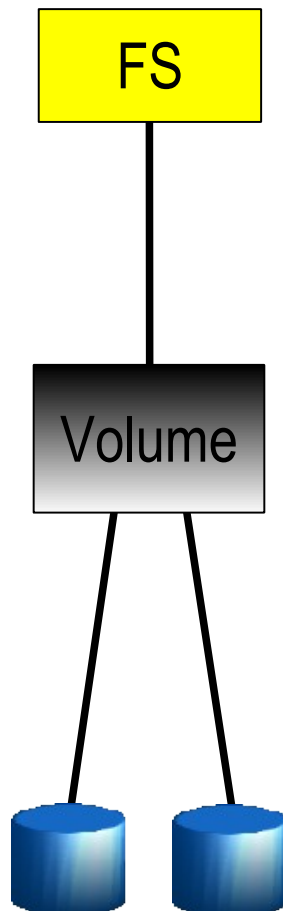
FS/Volume I/O Model

One-block-at-a-time I/O

- “Write this block, then that block, ...”
- Loss of power = loss of on-disk consistency
- Workaround: journaling, which is slow & complex

One-block-at-a-time I/O

- Write each block to each disk immediately to keep the mirrors in sync
- Loss of power = resync
- Slow, slow, slow



ZFS I/O Model

Object-based transactions

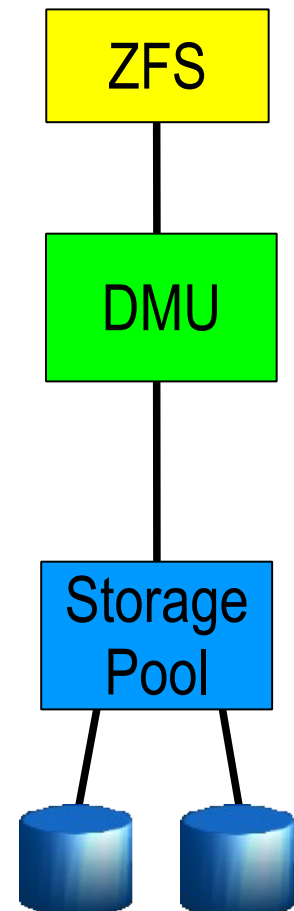
- “Make these 7 changes to these 3 objects”
- All-or-nothing

Transaction Group Commit

- Againing all-or-nothing
- Always consistent on disk
- No journal -- not needed

Transaction Group Batch I/O

- Schedule, aggregate, and issue I/O at will
- No resync if power lost
- As fast as the disk can go



Solaris ZFS

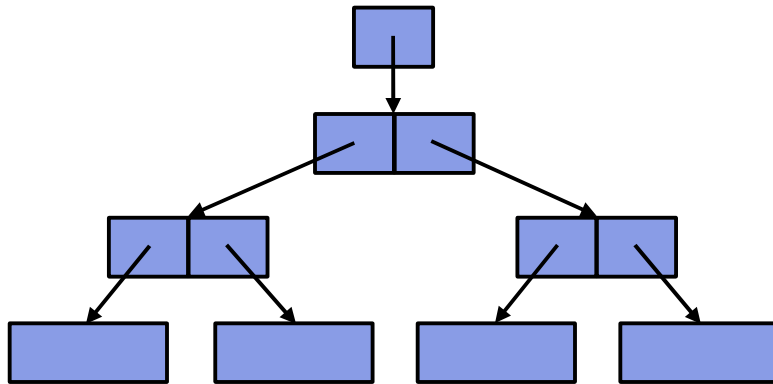
Data Integrity Model

- All operations are Copy-On-Write
 - > Never overwrite live data
 - > On-disk state always valid
 - > No need for fsck(1m)
- All operations are transactional
 - > Related changes succeed or fail as a whole
 - > No need for journaling
- All data is checksummed
 - > No silent data corruption
 - > No panics on bad metadata

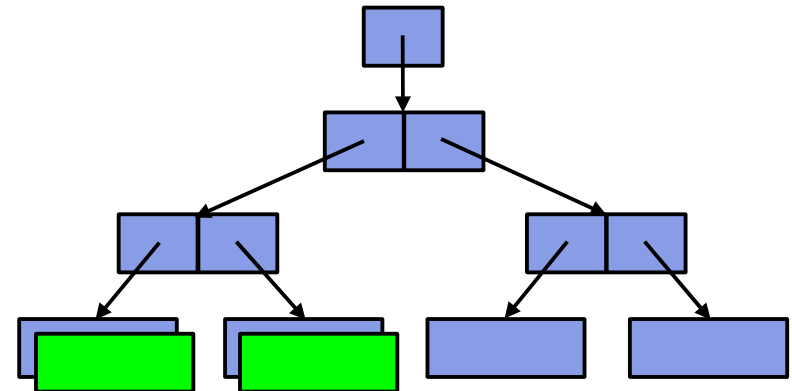
Solaris ZFS

Copy-On-Write transactions

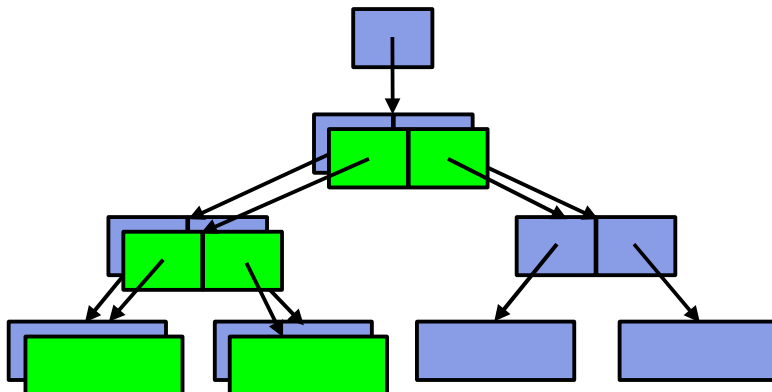
1. Initial block tree



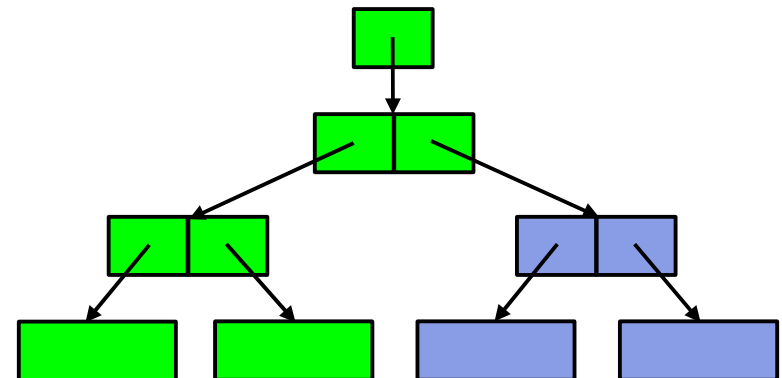
2. COW data block



3. COW indirect blocks



4. Rewrite uberblock (atomic)

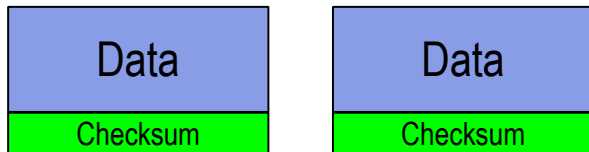


Solaris ZFS

End-to-End Checksums

Traditional checksums

- Checksum stored with data block
- Any self-consistent block will pass
- Best you can do w/o integrated stack
- Fundamental FS/volume limitation

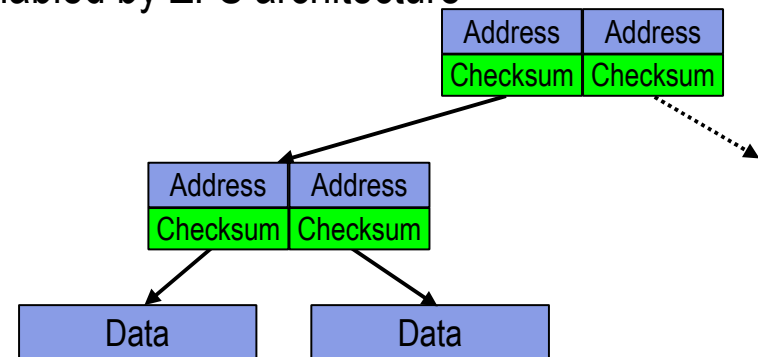


Only validates the media

- ✓ Bit rot
- ✗ Phantom writes
- ✗ Misdirected reads and writes
- ✗ DMA parity errors
- ✗ Driver bugs
- ✗ Accidental overwrite

ZFS checksums

- Checksum stored in ZFS block pointer
- Simple change with profound effects
- Entire pool is self-validating
- Enabled by ZFS architecture



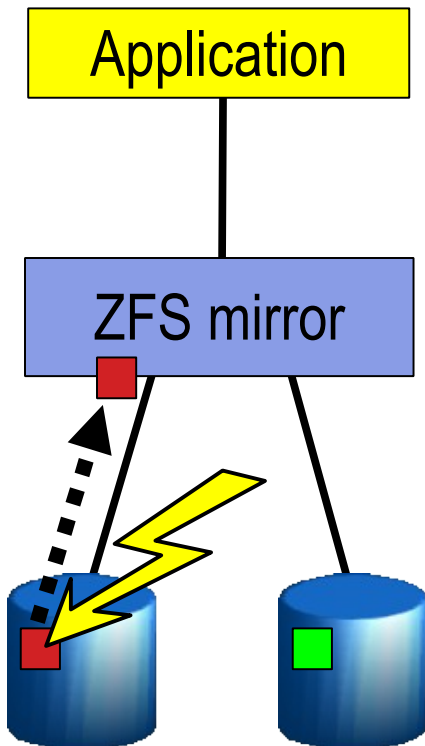
Validates the entire I/O path

- ✓ Bit rot
- ✓ Phantom writes
- ✓ Misdirected reads and writes
- ✓ DMA parity errors
- ✓ Driver bugs
- ✓ Accidental overwrite

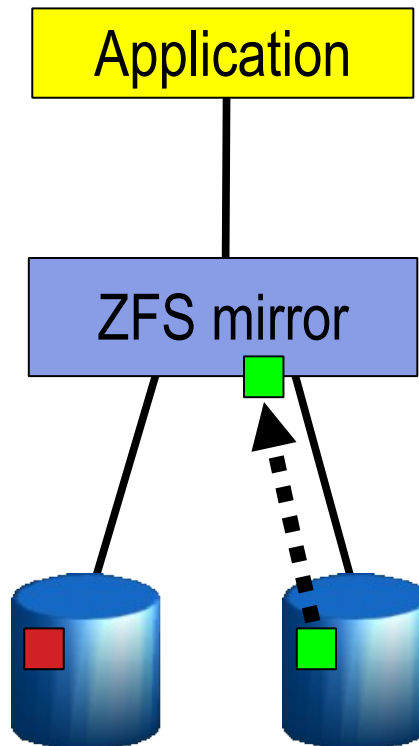
Solaris ZFS

Self-healing data

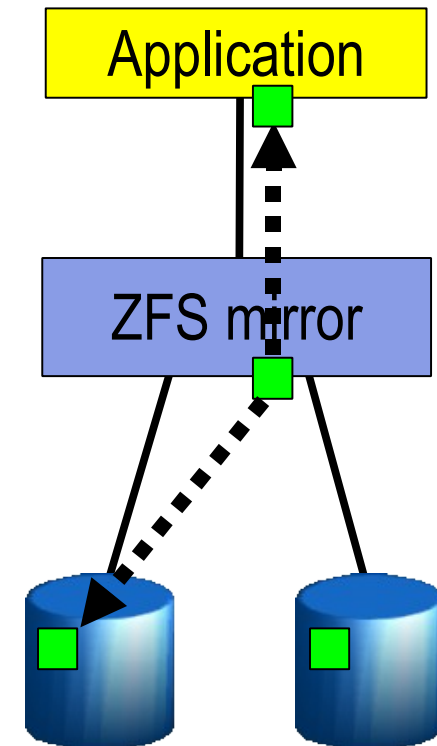
1. Application issues a read.
ZFS mirror tries first disk.
Checksum reveals that the
block is corrupt on disk.



2. ZFS tries second disk.
Checksum indicates that the
block is good.



3. ZFS returns good data to the
application and repairs the
damaged block.



Solaris ZFS

Storage Administration

- Pooled storage – no more volumes
 - > All storage is shared - no wasted space
 - > Grow and shrink are automatic
- Unlimited snapshots
 - > Read only snapshots: point-in-time copy of the data
 - > Read/write snapshots: multiple variants of the data
 - > Lets users recover lost data
- Host neutral on-disk format
 - > Adaptive endianness ensures neither platform pays a tax
- 100% online administration

Solaris ZFS

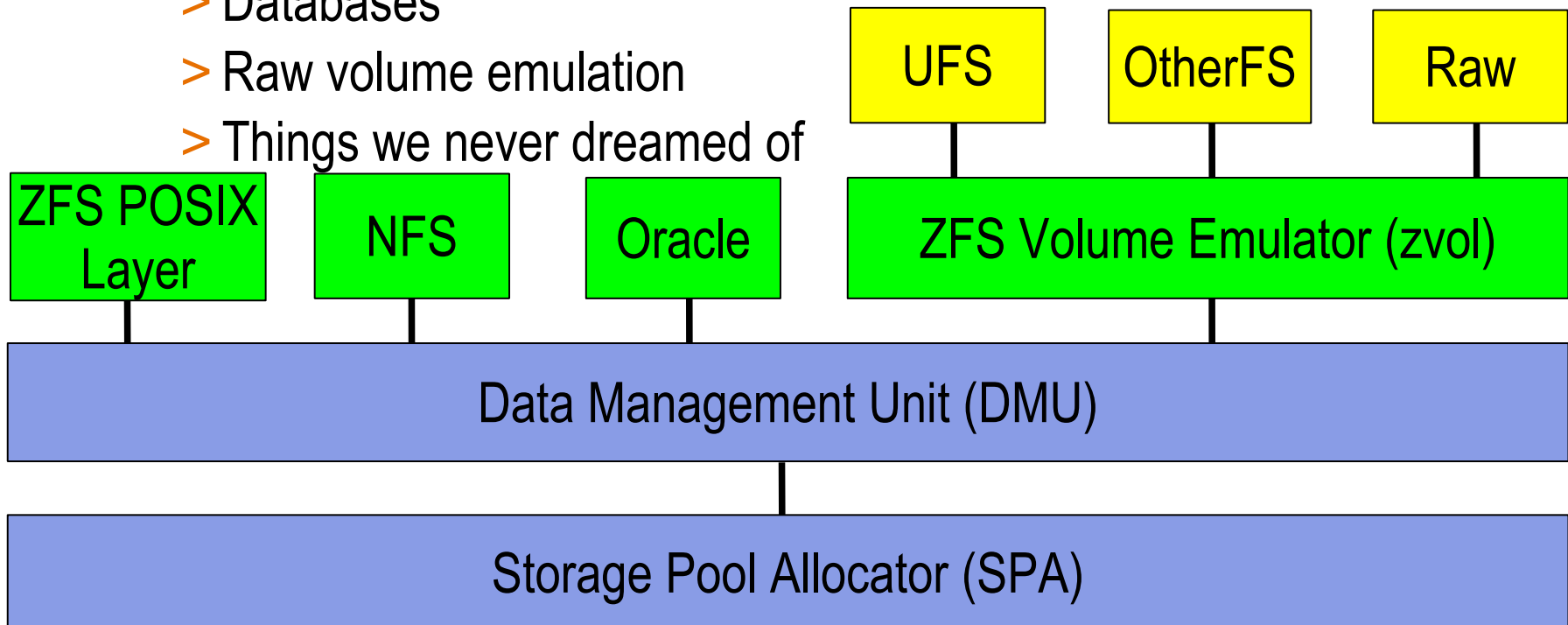
ZFS Administration

- Create a storage pool named “home”
`zpool create home mirror(disk1,disk2)`
- Create filesystem for “ann”, “bob”, “sue”
`zfs create home/ann /export/home/ann`
`zfs create home/bob /export/home/bob`
`zfs create home/sue /export/home/sue`
- Add more space to the “home” pool
`zpool add home mirror(disk3,disk4)`

Solaris ZFS

Object-Based Storage

- POSIX isn't the only game in town
 - > DMU provides a universal transactional object store
 - > Filesystems
 - > Databases
 - > Raw volume emulation
 - > Things we never dreamed of



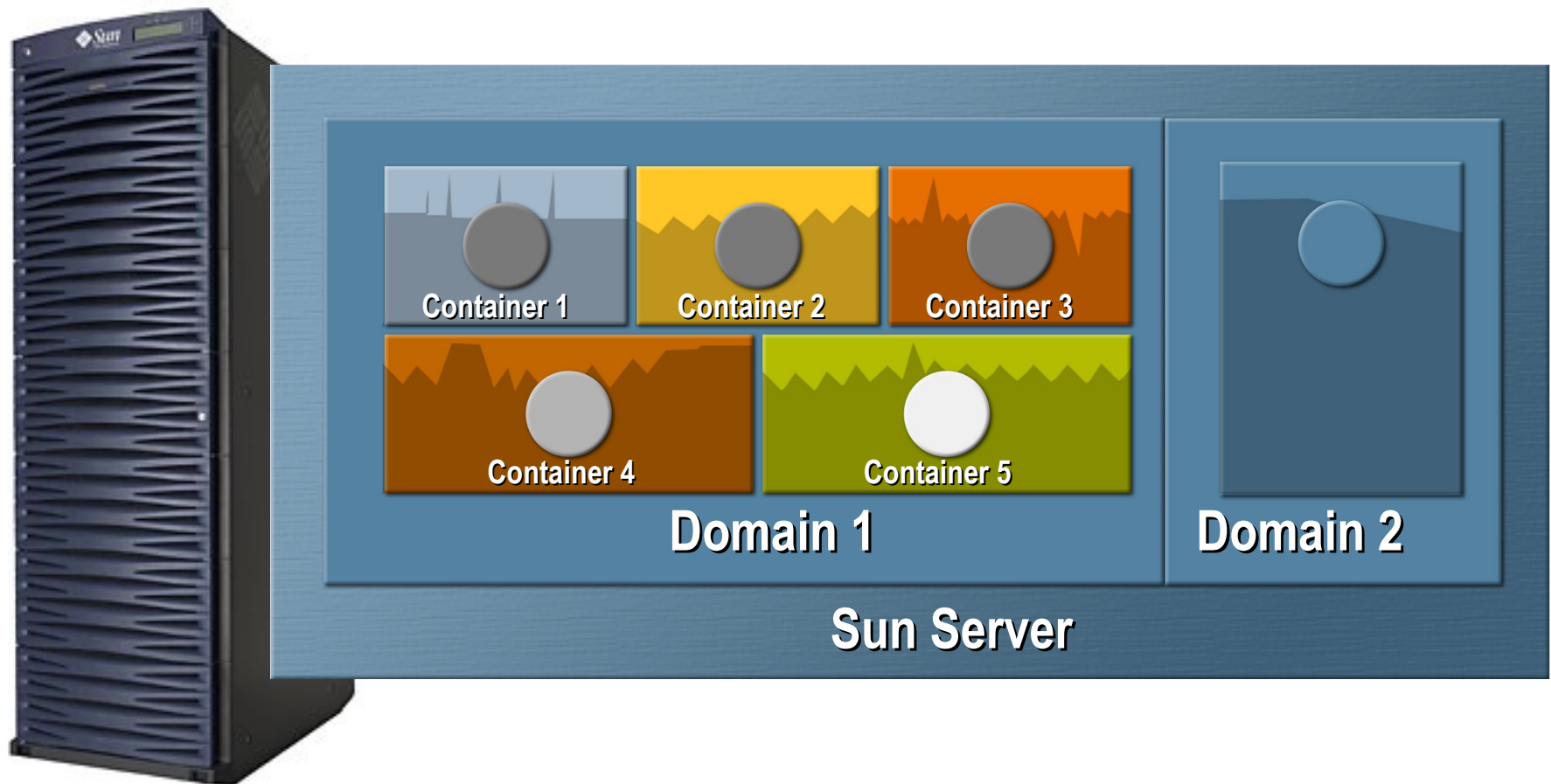
Solaris Containers

- Run multiple applications on one system
 - > Improve utilization by 4x
- Isolate applications from:
 - > Faults
 - > Intrusion
 - > Resource contention
- No performance hit



Solaris Containers

Server Virtualization



Solaris Containers

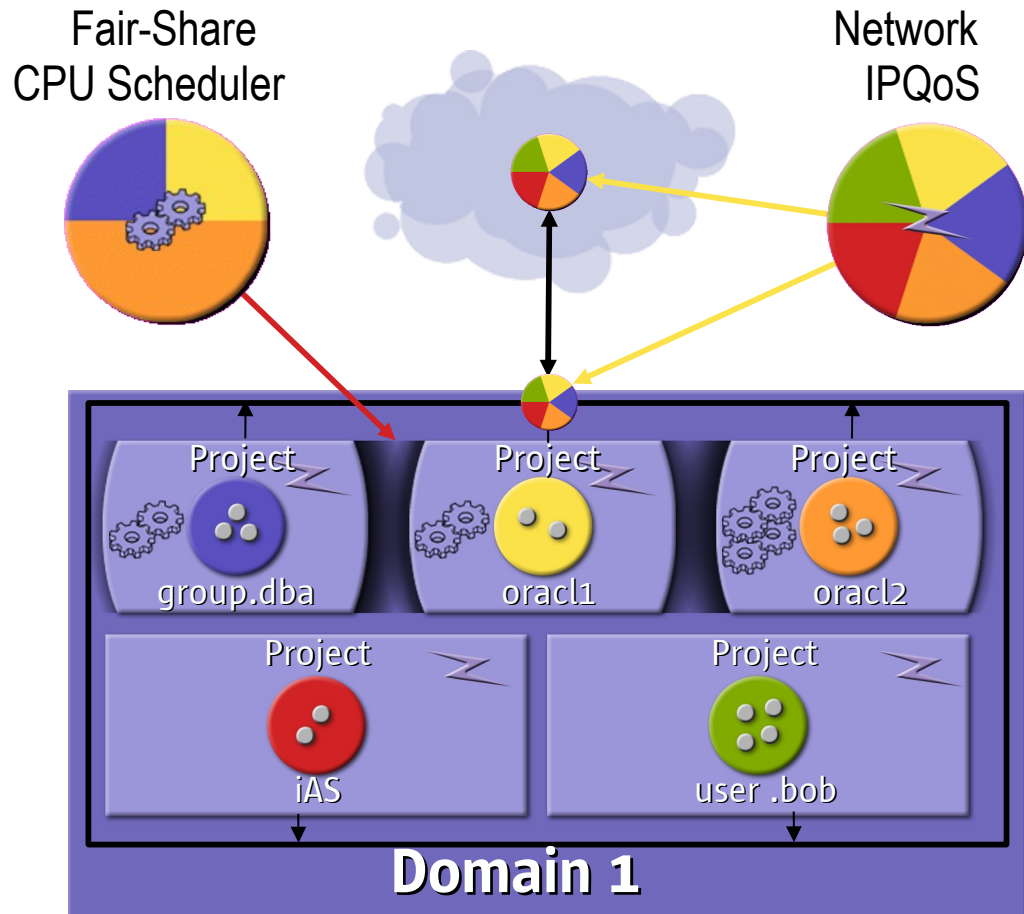
Components

- Full Resource Containment and control
 - > Provide predictable service levels
- Security isolation
 - > Prevent unauthorized access
- Fault isolation
 - > Minimize fault propagation and unplanned downtime
- Service Management Application
 - > Provide ease of management

Solaris Containers

Resource Management

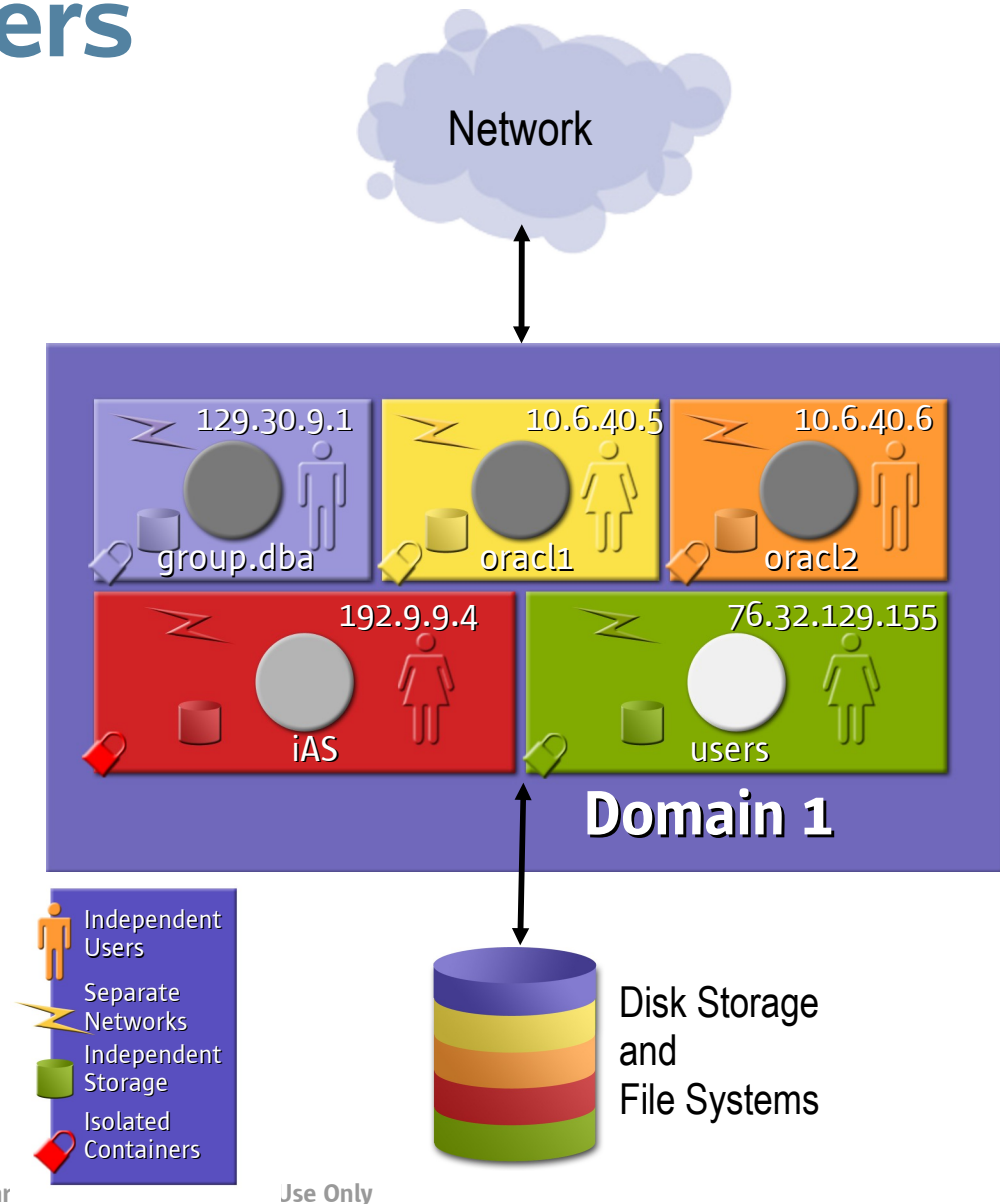
- Workload Metering
- Sub-Domain Partitioning
- Control CPU, Memory, and Network



Solaris Containers

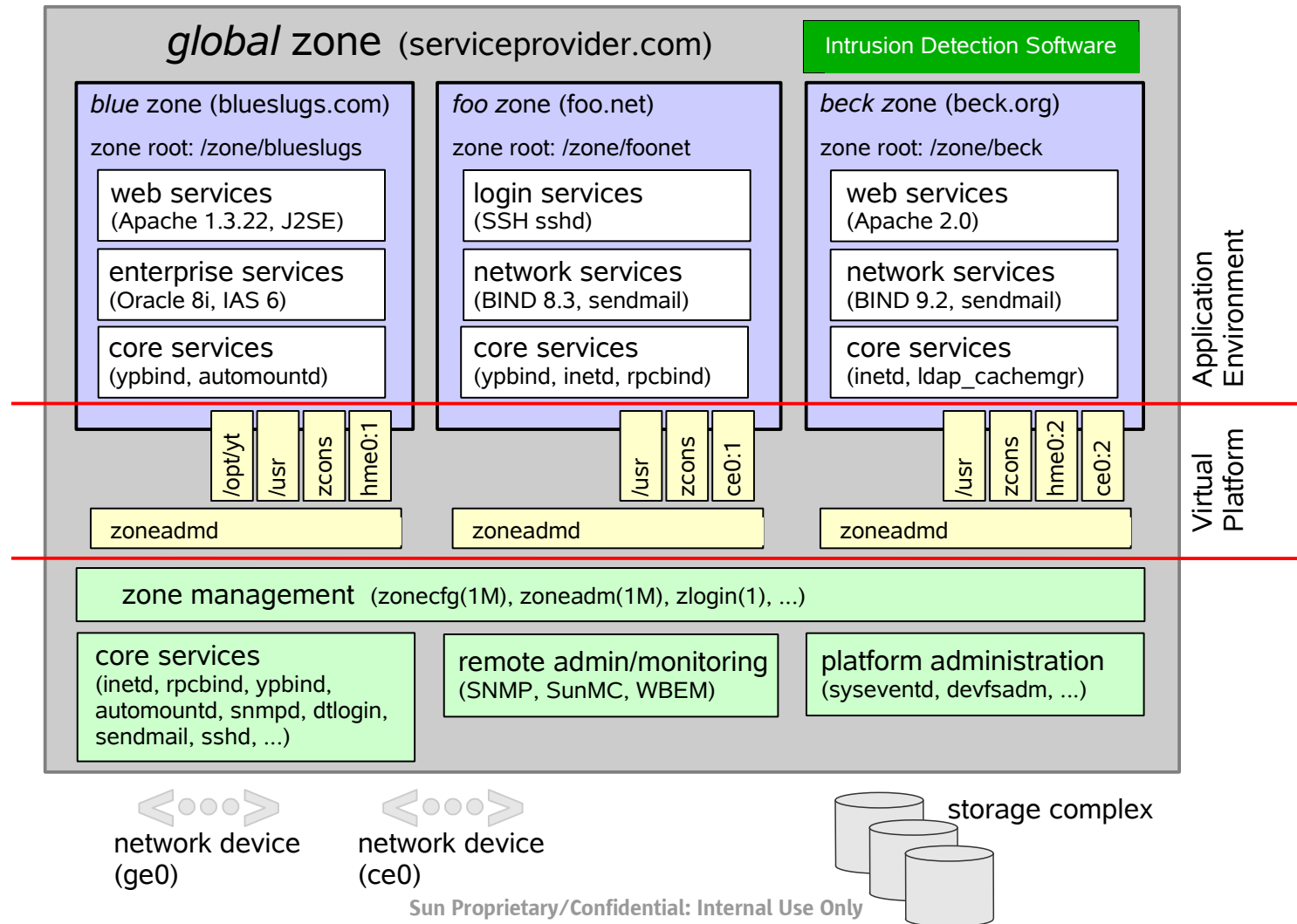
Solaris Zones

- Namespace isolation
- Virtualized OS
- Sharing for better utilization
- Application fault containment



Solaris Containers

Solaris Zones (2)



Solaris Containers

Solaris Zones (3)

- Separate namespaces
 - > users, root, IP addresses, ports, filesystem, ...
 - > Possibility for different name services (LDAP, ...)
- No visibility outside own environment
 - > ps, /proc, prstat, ... only show local processes
 - > No sharing memory other than local
- Filesystem is split
 - > Private piece of disk for identity & application
 - > Shared read-only to underlying OS (/lib,/usr,...)

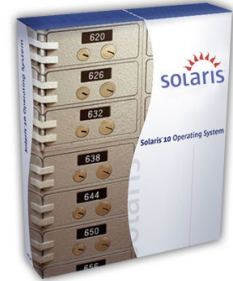
Solaris Containers

Solaris Zones (4)

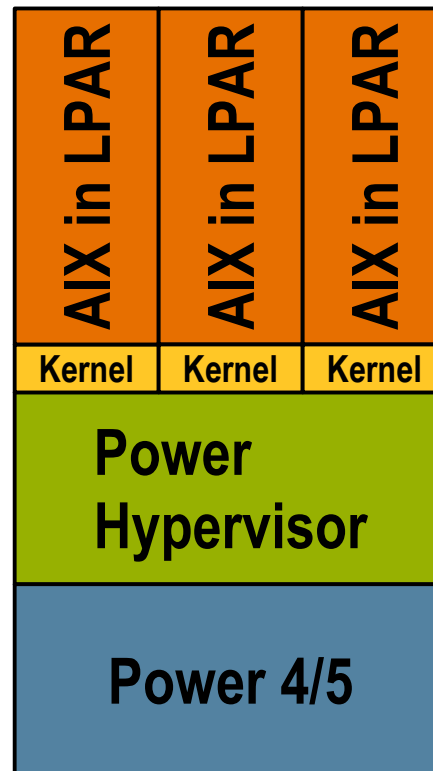
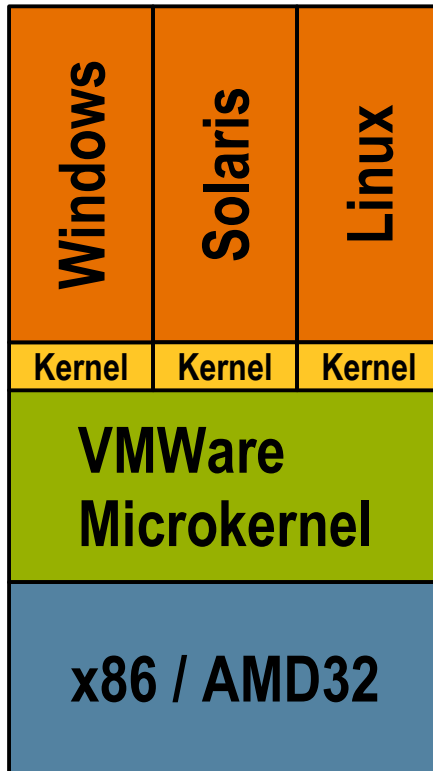
- Very scaleable
 - > Many (+4000) per OS instance
 - > Very very very low overhead
- Platform independent
 - > Runs on SPARC and x86 (also Opteron)
 - > No need for special HW firmware support
- Hardware is dynamically shared
 - > Idle cycles can be used by others
 - > Uses the RM for guarantees

Solaris Containers

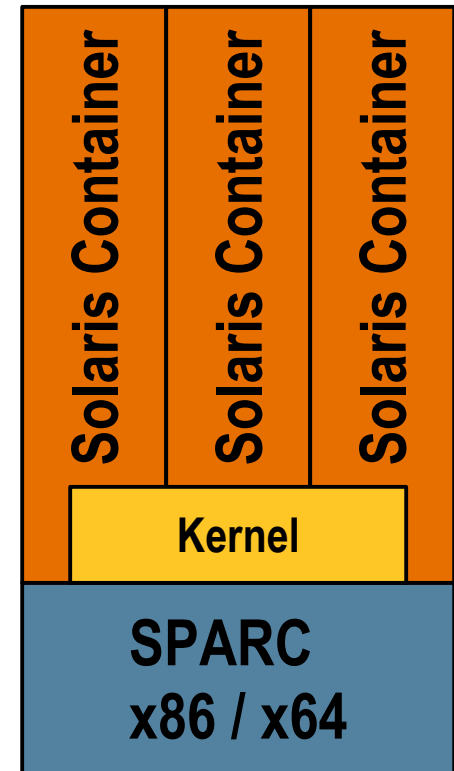
vs. Other Virtualization Technologies



Optimal Utilization



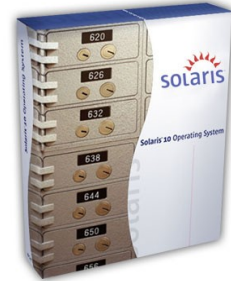
IBM is a Proprietary, Confidential, Internal Use Only



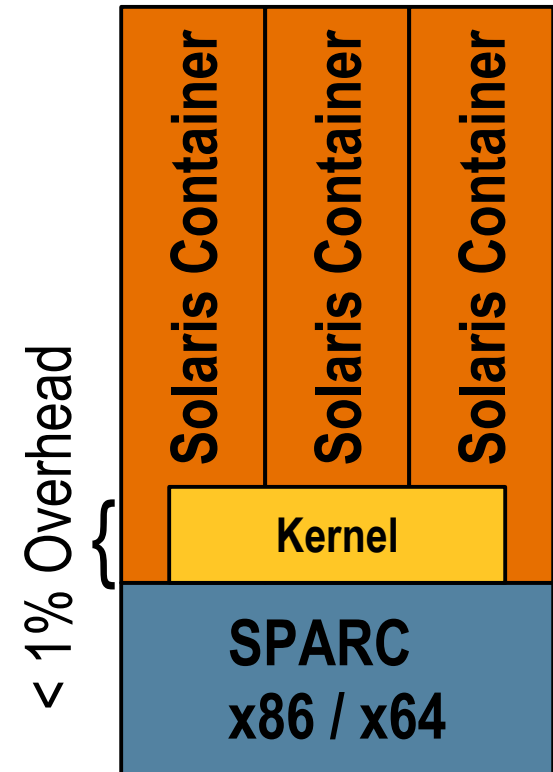
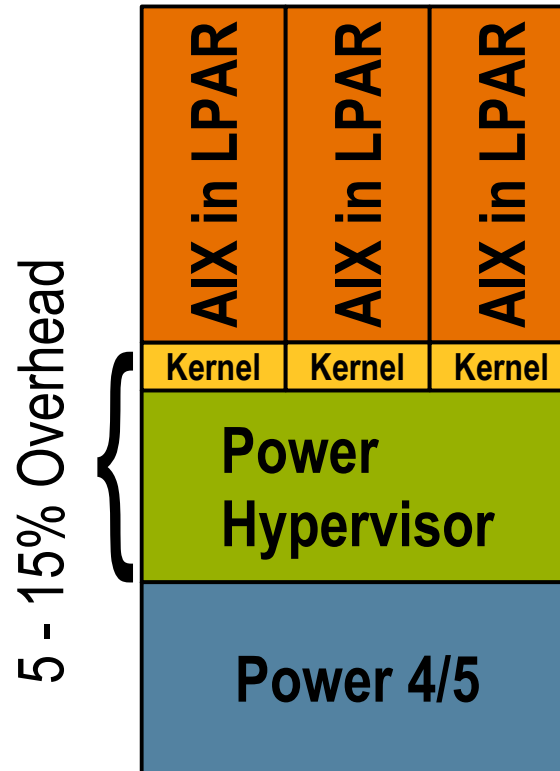
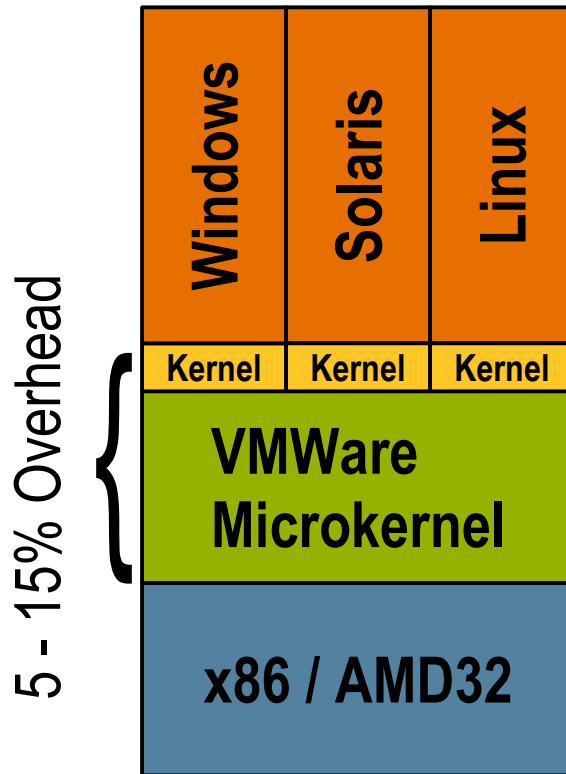
Solaris Containers

vs. Other Virtualization Technologies

- Compare observability, manageability, performance, cost, platform choice



Optimal Utilization



Solaris 10: A Generation Ahead



Extreme Performance



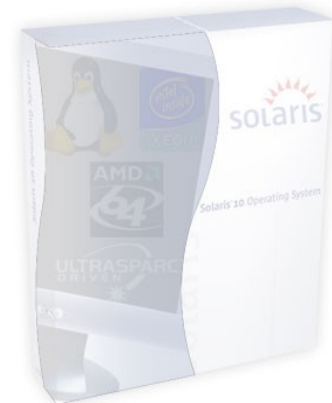
Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

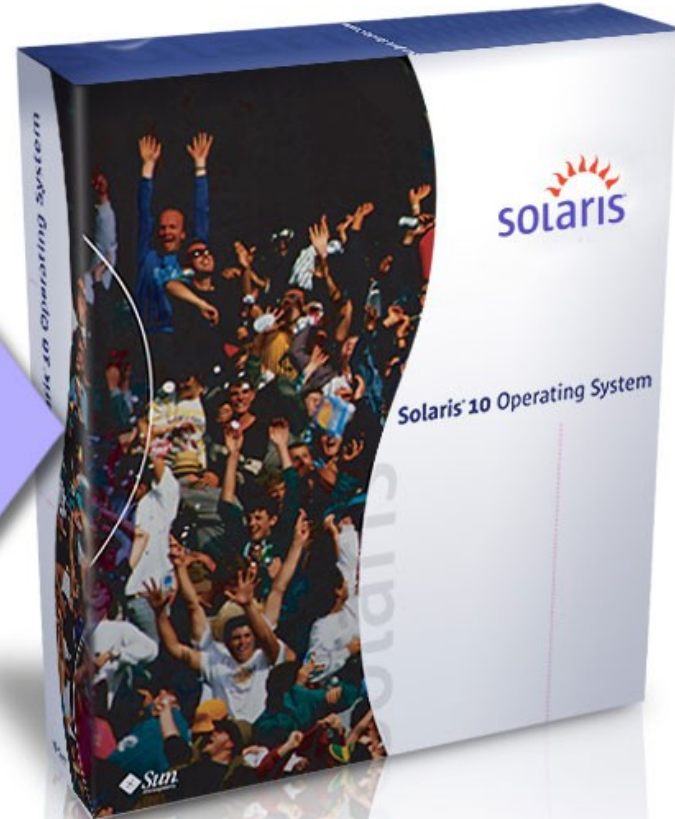
Military Grade Security



Solaris



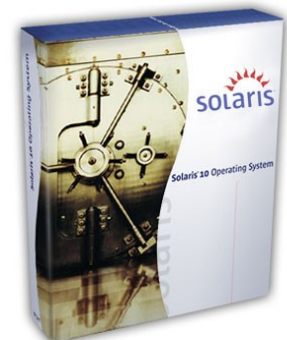
**Trusted
Solaris**



Proven Security

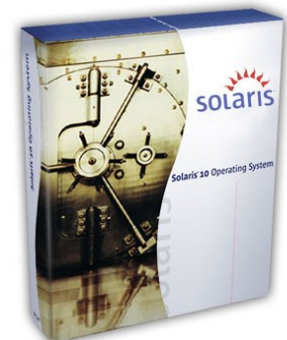
Over 20 Years of Design, Testing, Refinement and Experience

- Administrative
 - > Secure out of the box
 - > System integrity
 - > User rights management
 - > Containers
- Application
 - > Process rights management
 - > Cryptographic framework
- Network
 - > IP filtering



Process Rights Management

- Root User (UID 0) has no “inalienable rights”
 - > But traditional root privileges enabled for compatibility
- Granularity: 40+ privileges associated with users, processes
 - > SetUID becomes deprecated practice
- Selectable privilege inheritance
- Full compatibility with previous environments, applications



Process Rights Management

Privilege Sets

- Effective set
 - > Privileges currently in effect
 - > Privileges can be added or dropped
- Permitted Set
 - > Upper bound on Effective Set for this process
 - > Privileges can be dropped (changes Effective)

Process Rights Management

Privilege Sets (2)

- Inheritable Set
 - > Default privileges given to child processes
 - > Becomes child's Permitted and Effective Set
- Limit Set
 - > Upper bound for Inheritable Set
 - > Typically contains all privileges

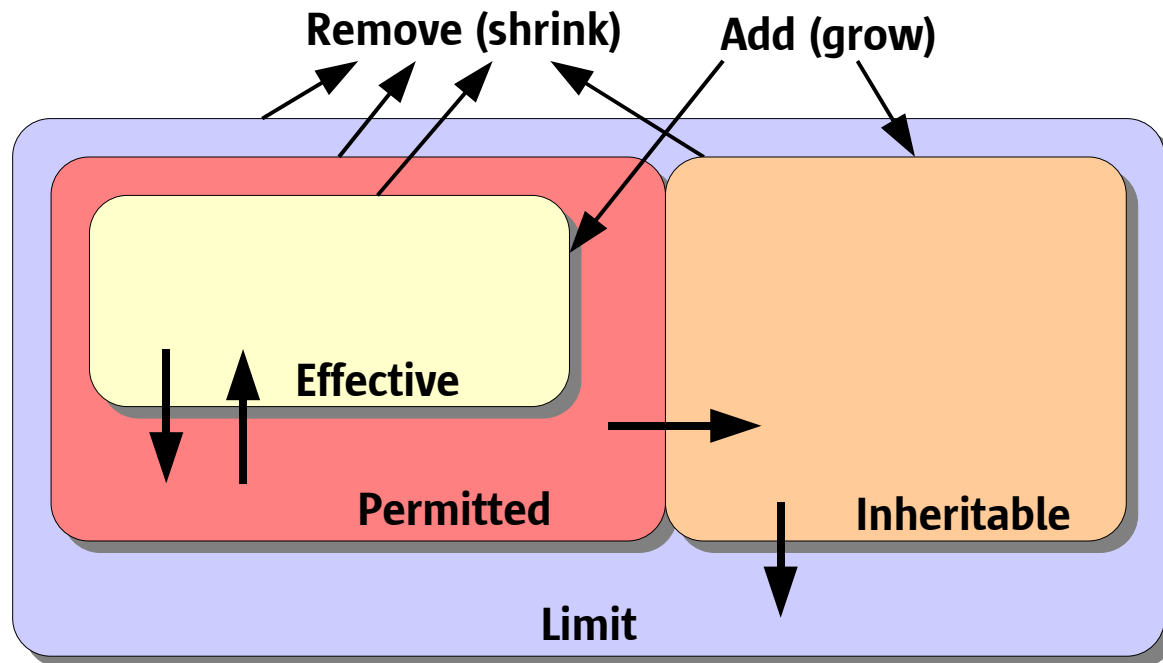
Process Rights Management

Allowed Privilege Set Changes

- Add privilege to:
 - > Permitted (P)
 - > Effective (E)
 - > Inheritable (I)
- Remove privilege from any set
 - > Privileges removed from P are automatically removed from E
 - > Privileges removed from L are *not* removed from I, E, P until exec time

Process Rights Management

Allowed Privilege Set Changes



Solaris 10's 48 Privilege Names

"contract_event"	Request reliable delivery of events	"proc_fork"	Allow use of fork*() calls
"contract_observer"	Observe contract events for other users	"proc_info"	Examine /proc of other processes
"cpc_cpu"	Access to per-CPU perf counters	"proc_lock_memory"	Lock pages in physical memory
"dtrace_kernel"	DTrace kernel tracing	"proc_owner"	See/modify other process states
"dtrace_proc"	DTrace process-level tracing	"proc_prioctl"	Increase priority/sched class
"dtrace_user"	DTrace user-level tracing	"proc_session"	Signal/trace other session process
"file_chown"	Change file's owner/group IDs	"proc_setid"	Set process UID
"file_chown_self"	Give away (chown) files	"proc_taskid"	Assign new task ID
"file_dac_execute"	Override file's execute perms	"proc_zone"	Signal/trace processes in other zones
"file_dac_read"	Override file's read perms	"sys_acct"	Manage accounting system (acct)
"file_dac_search"	Override dir's search perms	"sys_admin"	System admin tasks (e.g. domain name)
"file_dac_write"	Override (non-root) file's write perms	"sys_audit"	Control audit system
"file_link_any"	Create hard links to diff uid files	"sys_config"	Manage swap
"file_owner"	Non-owner can do misc owner ops	"sys_devices"	Override device restricts (exclusive)
"file_setid"	Set uid/gid (non-root) to diff id	"sys_ipc_config"	Increase IPC queue
"ipc_dac_read"	Override read on IPC/Shared Mem perms	"sys_linkdir"	Link/unlink directories
"ipc_dac_write"	Override write on IPC/Shared Mem perms	"sys_mount"	Filesystem admin (mount,quota)
"ipc_owner"	Override set perms/owner on IPC	"sys_net_config"	Config net interfaces,routes,stack
"net_icmpaccess"	Send/Receive ICMP packets	"sys_nfs"	Bind NFS ports and use syscalls
"net_privaddr"	Bind to privilege port (<1023+extras)	"sys_res_config"	Admin processor sets, res pools
"net_rawaccess"	Raw access to IP	"sys_resource"	Modify res limits (rlimit)
"proc_audit"	Generate audit records	"sys_suser_compat"	3rd party modules use of suser
"proc_chroot"	Change root (chroot)	"sys_time"	Change system time
"proc_clock_highres"	Allow use of hi-res timers		
"proc_exec"	Allow use of execve()		

Interesting
Basic

Some interesting privileges
Non-root privileges

Solaris 10's 48 Privilege Names

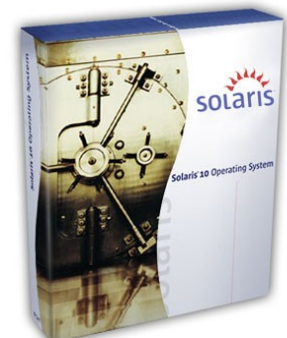
"contract_event"	Request reliable delivery of events	"proc_fork"	Allow use of fork*() calls
"contract_observer"	Observe contract events for other users	"proc_info"	Examine /proc of other processes
"cpc_cpu"	Access to per-CPU perf counters	"proc_lock_memory"	Lock pages in physical memory
"dtrace_kernel"	DTrace kernel tracing	"proc_owner"	See/modify other process states
"dtrace_proc"	DTrace process-level tracing	"proc_priocntl"	Increase priority/sched class
"dtrace_user"	DTrace user-level tracing	"proc_session"	Signal/trace other session process
"file_chown"	Change file's owner/group IDs	"proc_setid"	Set process UID
"file_chown_self"	Give away (chown) files	"proc_taskid"	Assign new task ID
"file_dac_execute"	Override file's execute perms	"proc_zone"	Signal/trace processes in other zones
"file_dac_read"	Override file's read perms	"sys_acct"	Manage accounting system (acct)
"file_dac_search"	Override dir's search perms	"sys_admin"	System admin tasks (e.g. domain name)
"file_dac_write"	Override (non-root) file's write perms	"sys_audit"	Control audit system
"file_link_any"	Create hard links to diff uid files	"sys_config"	Manage swap
"file_owner"	Non-owner can do misc owner ops	"sys_devices"	Override device restricts (exclusive)
"file_setid"	Set uid/gid (non-root) to diff id	"sys_ipc_config"	Increase IPC queue
"ipc_dac_read"	Override read on IPC/Shared Mem perms	"sys_linkdir"	Link/unlink directories
"ipc_dac_write"	Override write on IPC/Shared Mem perms	"sys_mount"	Filesystem admin (mount,quota)
"ipc_owner"	Override set perms/owner on IPC	"sys_net_config"	Config net interfaces,routes,stack
"net_icmpaccess"	Send/Receive ICMP packets	"sys_nfs"	Bind NFS ports and use syscalls
"net_privaddr"	Bind to privilege port (<1023+extras)	"sys_res_config"	Admin processor sets, res pools
"net_rawaccess"	Raw access to IP	"sys_resource"	Modify res limits (rlimit)
"proc_audit"	Generate audit records	"sys_suser_compat"	3rd party modules use of suser
"proc_chroot"	Change root (chroot)	"sys_time"	Change system time
"proc_clock_highres"	Allow use of hi-res timers		
"proc_exec"	Allow use of execve()		

Interesting
Basic
Zones

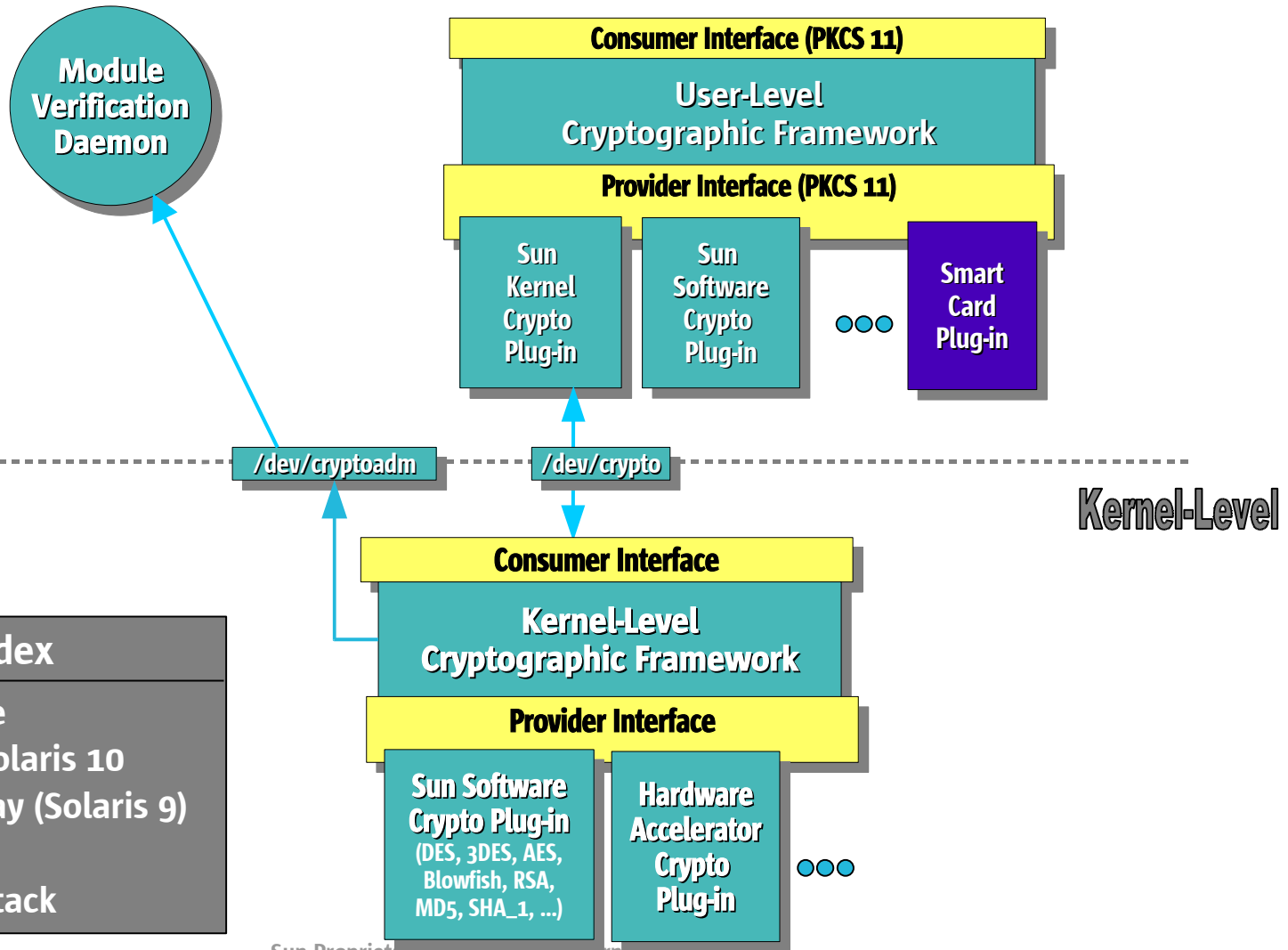
Some interesting privileges
Non-root privileges
Privileges removed from Zones

Cryptographic Framework

- Extensible cryptographic interfaces
 - > A common framework for both providing and using cryptographic functionality
 - > A common interface for cryptographic functions whether completed in hardware or software
- Supports major algorithms
 - > encrypt, hash etc
 - > AES, DES, 3DES, RSA, SHA-1, MD5 etc
 - > Optimized for both SPARC and x86



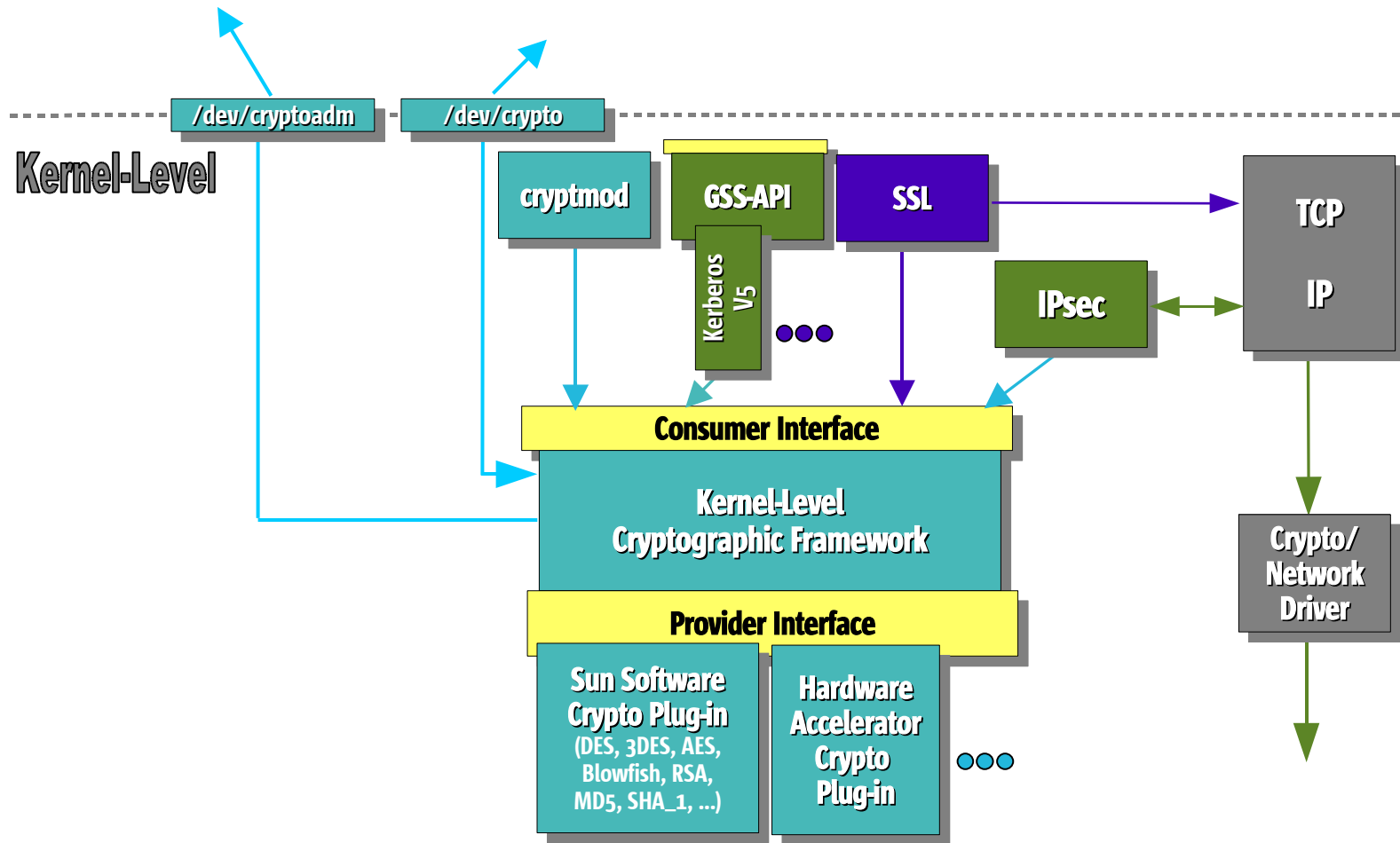
Cryptographic Framework



Color Index	
	Future release
	Expected in Solaris 10
	Available today (Solaris 9)
	ISV Interfaces
	Networking stack

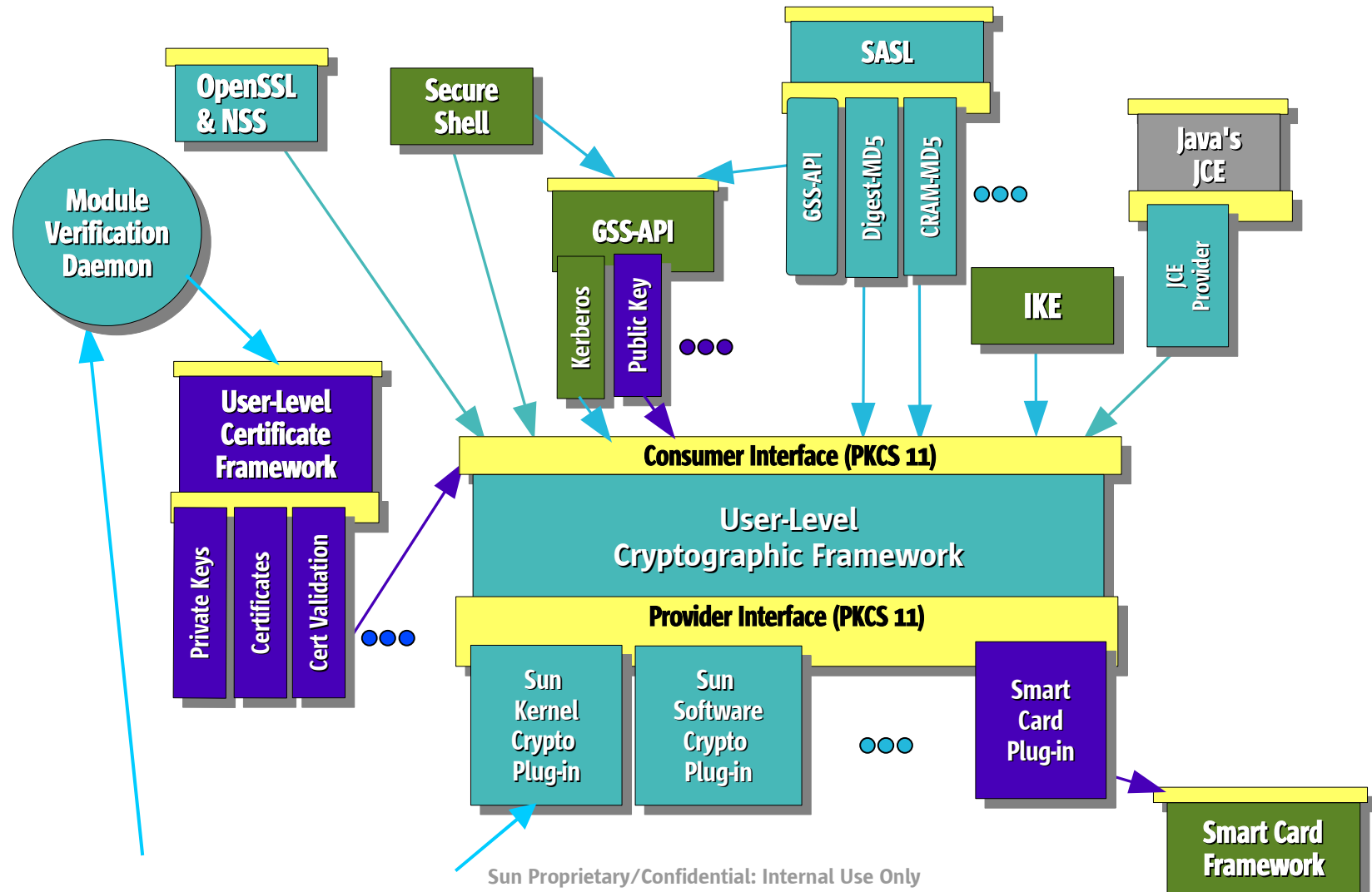
Cryptographic Framework

Kernel-level Components



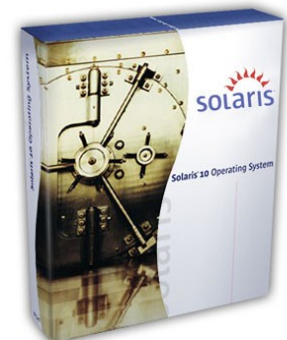
Cryptographic Framework

User-level Components



IP Filter

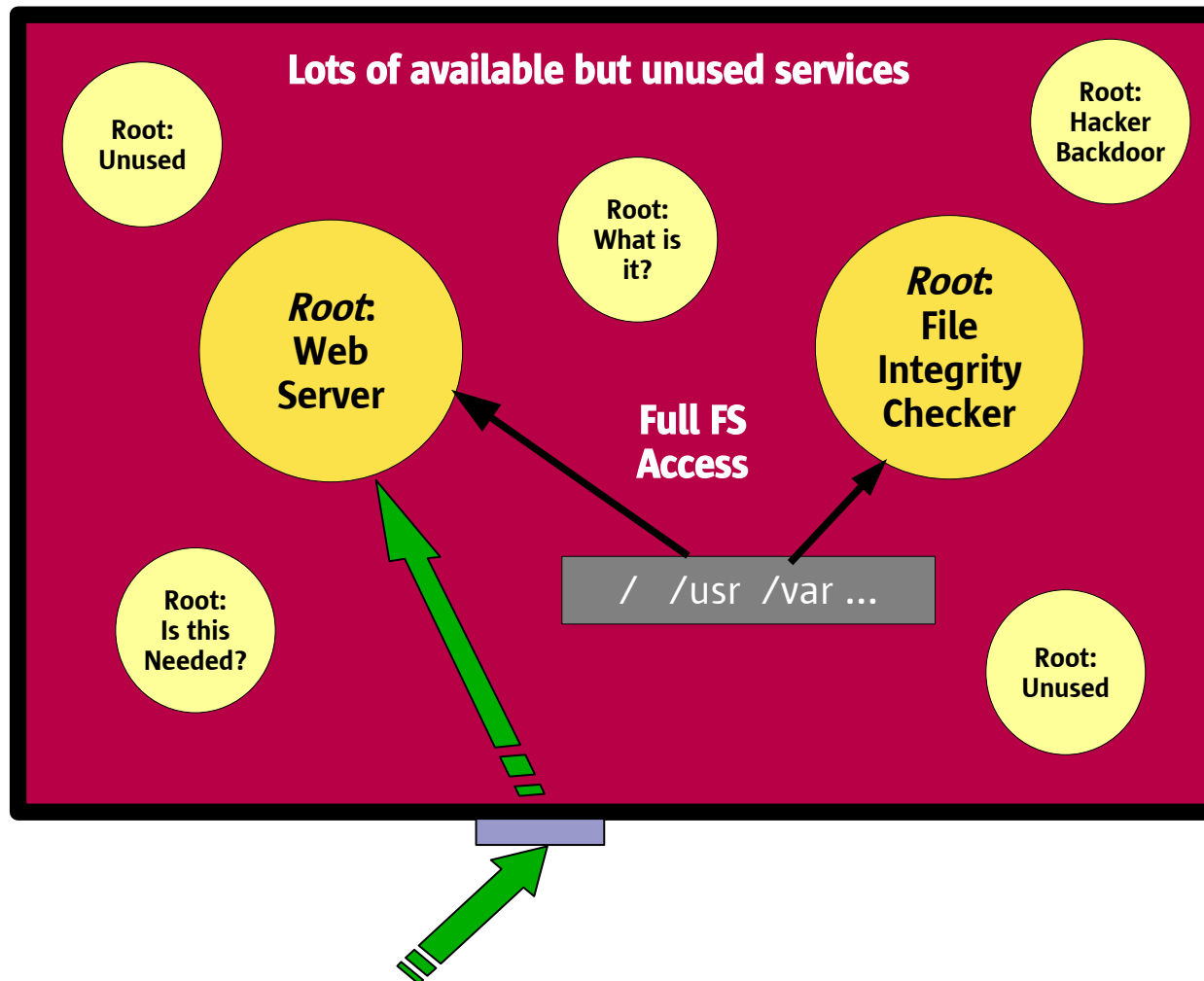
- Stateful and stateless packet inspection
- Kernel-based packet filtering
- Protocol proxies (TCP, UDP, FTP, rcmds ...)
- Text-based configuration
- Support for both NAT and PAT
- Logging
- Small footprint
- Minimal requirements



Integrity Checking (BART)

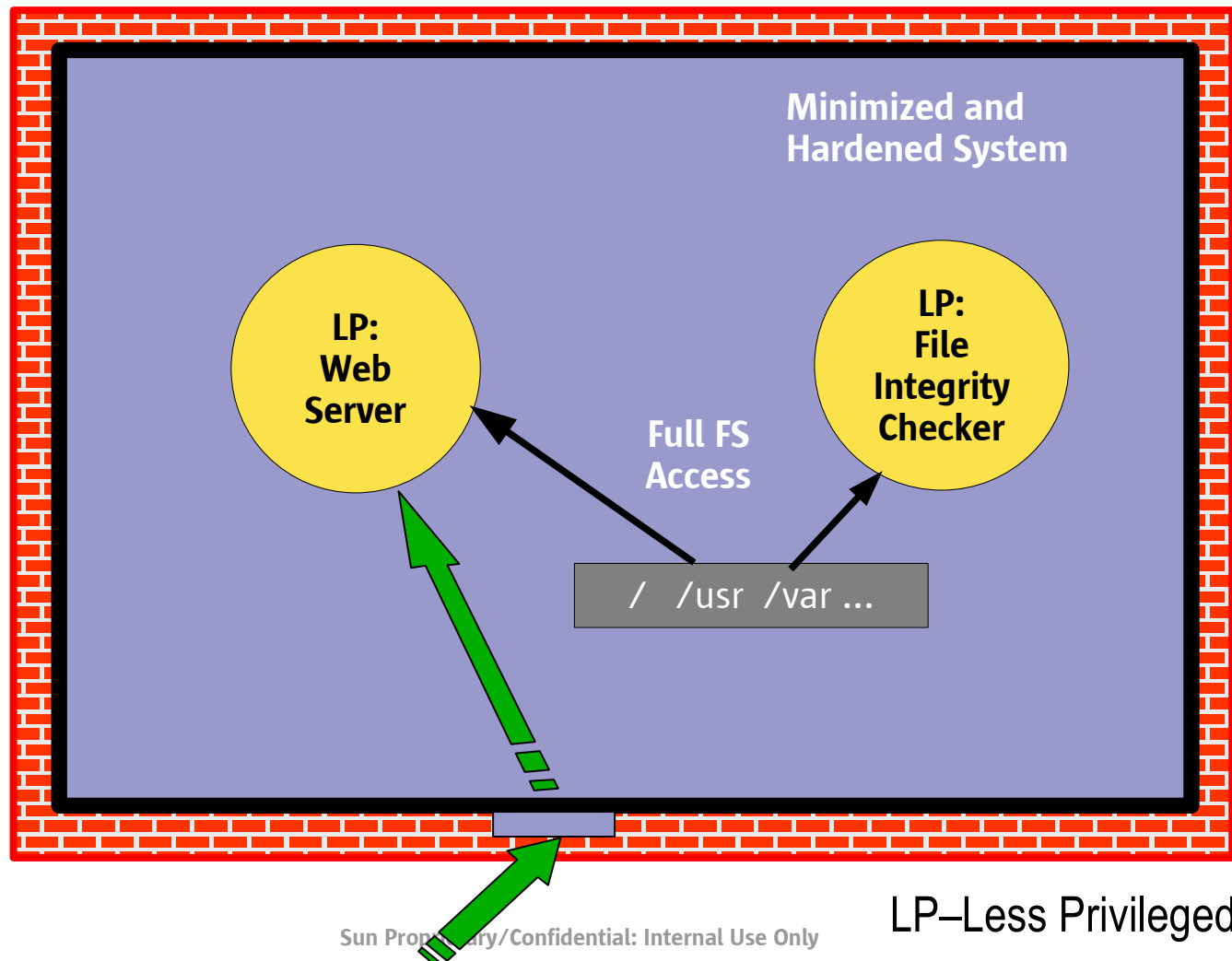
- Basic Audit and Reporting Tool
- Intrusion detection
- Verifies configuration files
- Look for unauthorized modifications
 - > Size, owner, mode, ...
 - > MD5 checksum

Today's Typical Deployed System



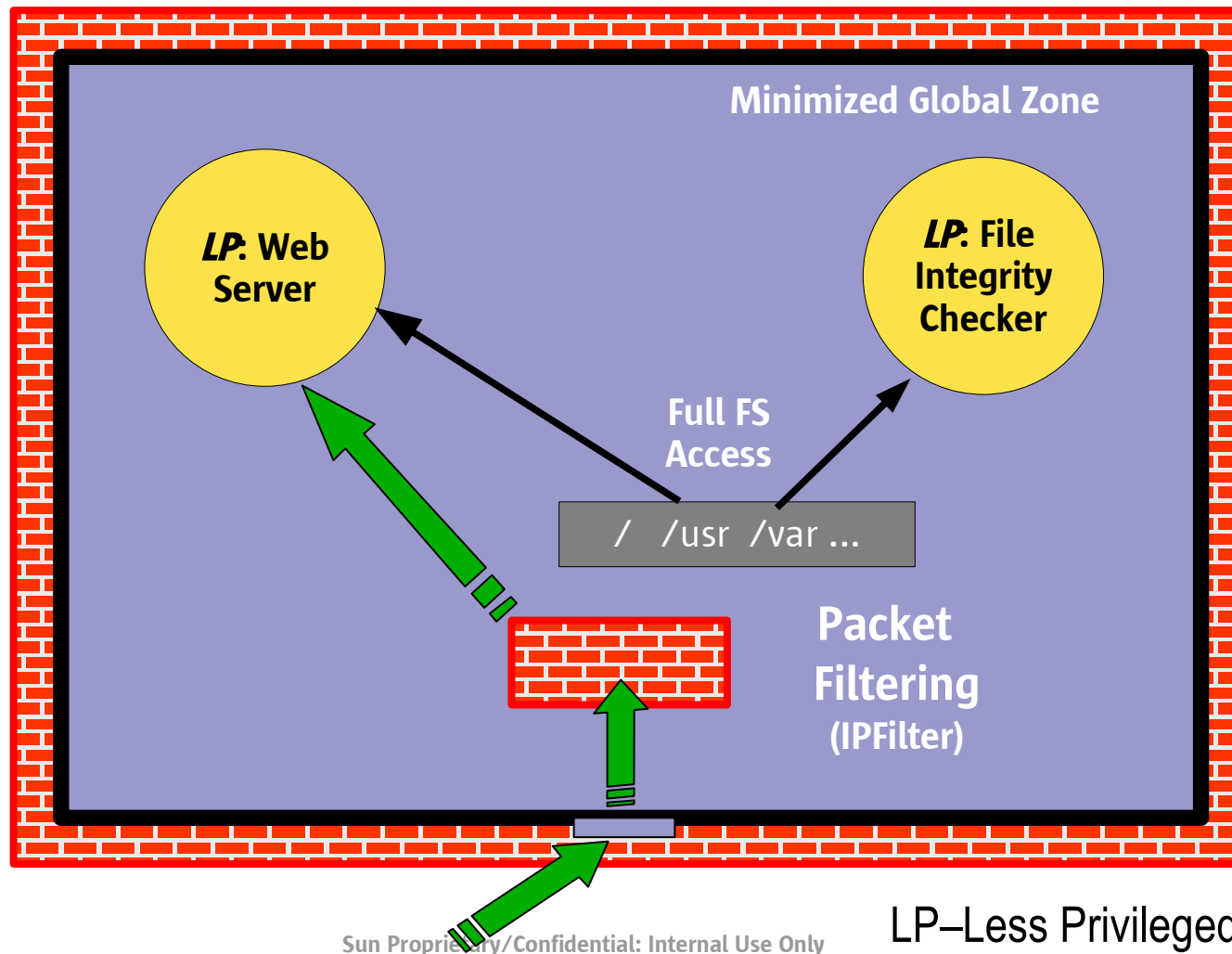
Solaris 10 Protected System

OS Hardening



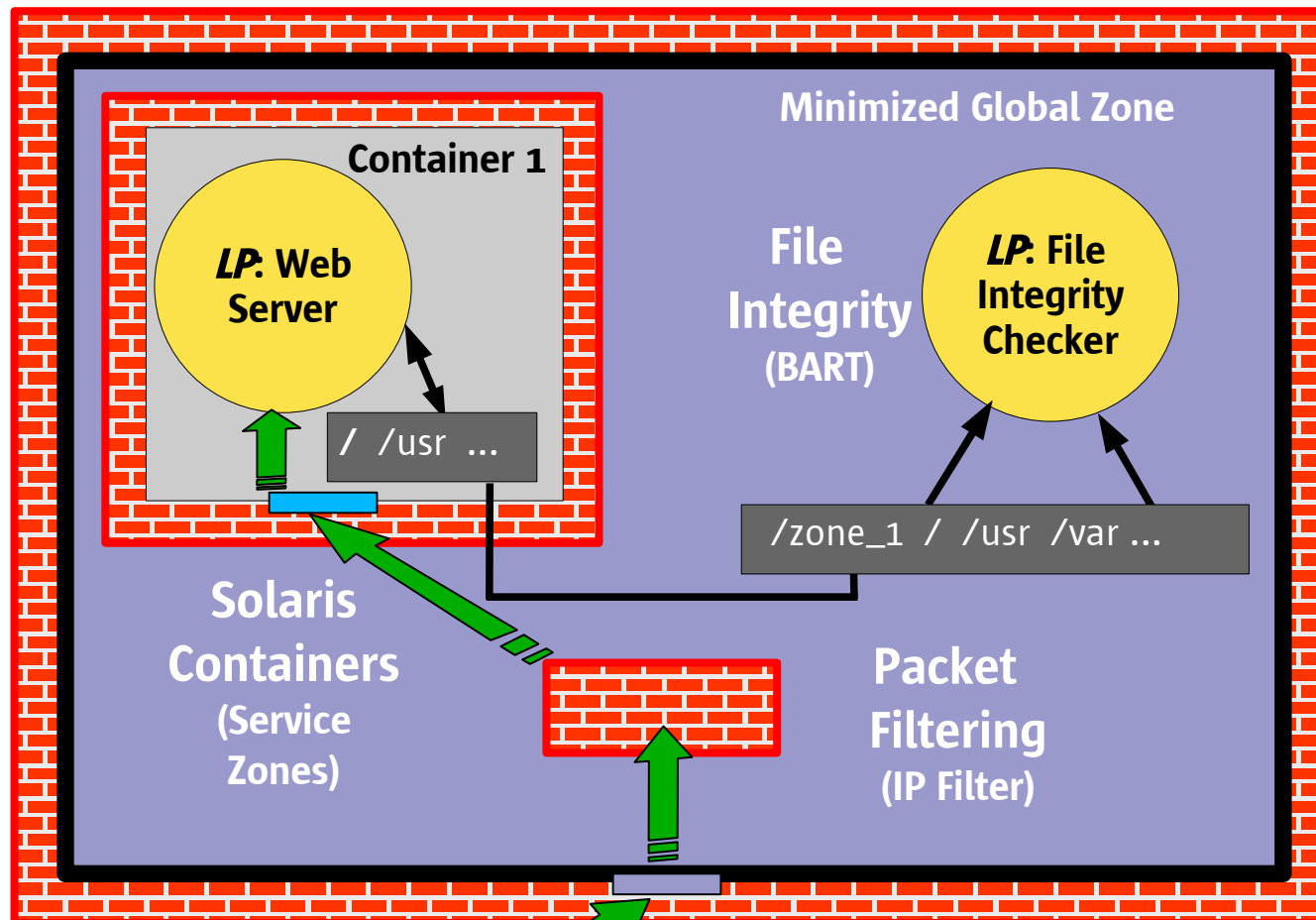
Solaris 10 Protected System

Packet Filtering



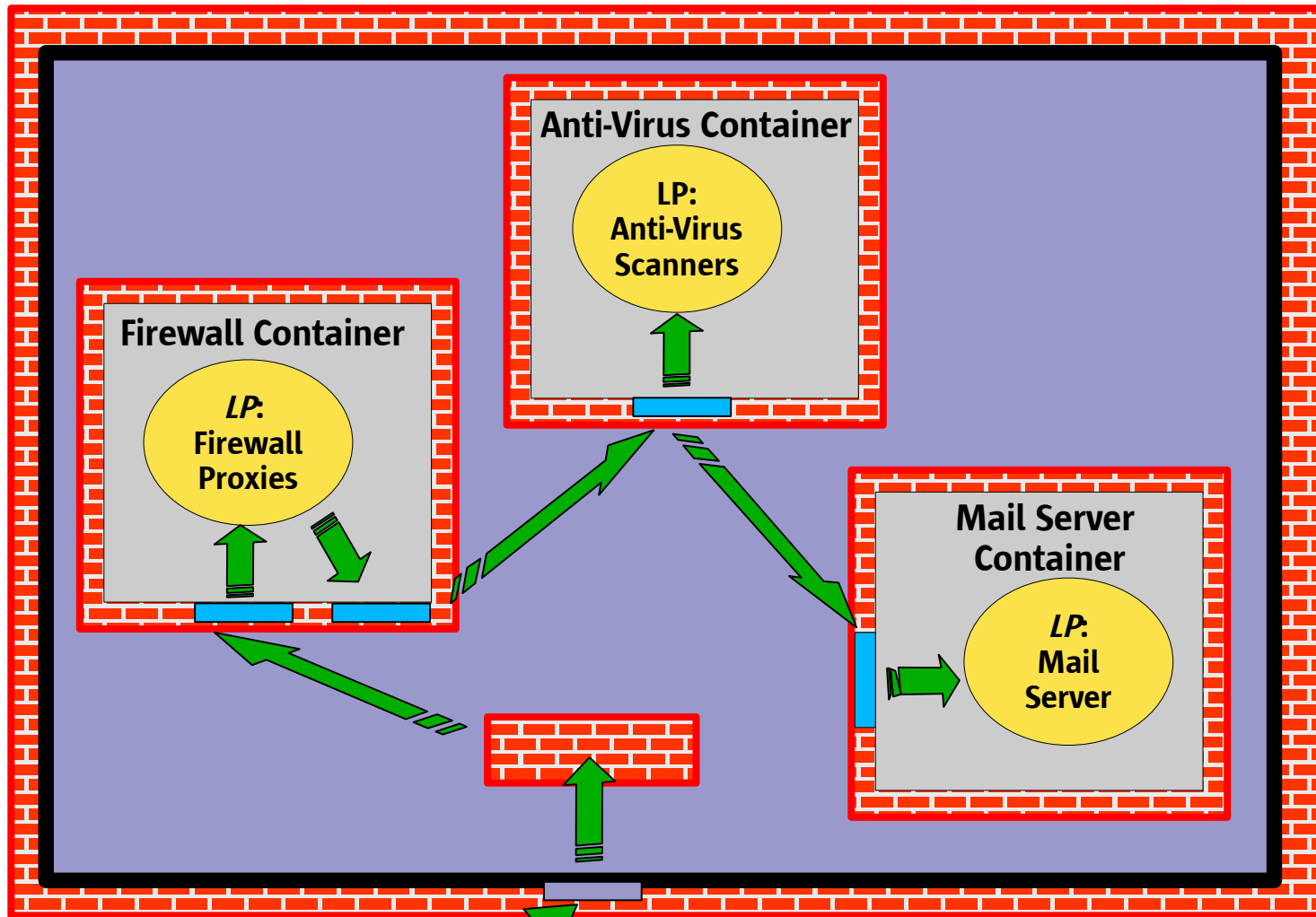
Solaris 10 Protected System

Solaris Containers and BART



Solaris 10 Protected System

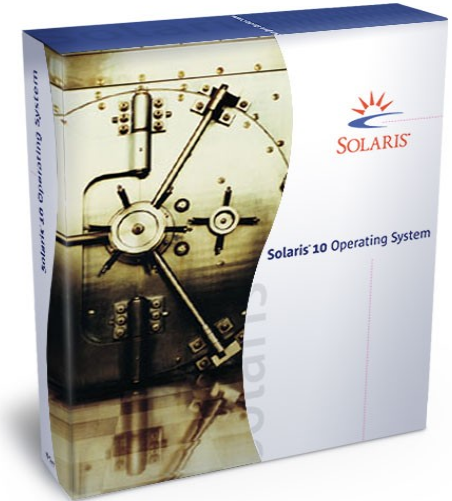
Multi-Tier System



Trusted Solaris OS

Military-Grade Security, Enterprise-Ready

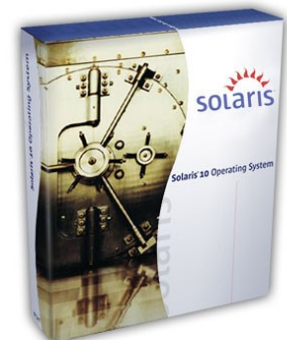
- Unmatched certification
 - > EAL4 w/ LSPP, RBACPP, CAPP
- Highest level of privacy, accountability
 - > Least privilege implementation
 - > Data labels combined with RBAC
 - > Audit trails integrated with labeling
- Ideal for regulatory compliance
 - > Sarbanes Oxley, GLBA, HIPAA, EU Data privacy



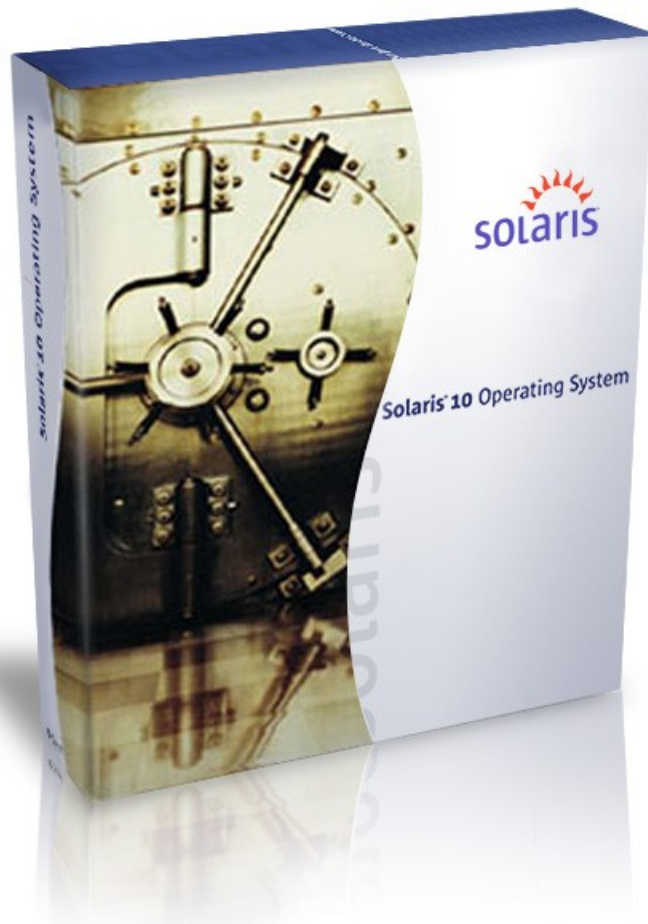
Solaris Trusted Extensions

New Paradigm for Solaris 10—Available Q3CY06

- Desktop support
- Distributed role-based access control
- Labeled zones
- Labeled mount points
- Multilevel networking
- Multilevel desktop
- Labeled printing
- Labeled administration



Multi-Level Labeled Security



Solaris Trusted Extensions

Adds labeled security to Solaris 10

Multi-level networking, printing

Multi-level CDE & Java DS* GUI

Leverages User & Process RM

Uses Containers

Compatible with all Solaris apps

Target of CAPP, RBACPP, LSPP
@ EAL 4+

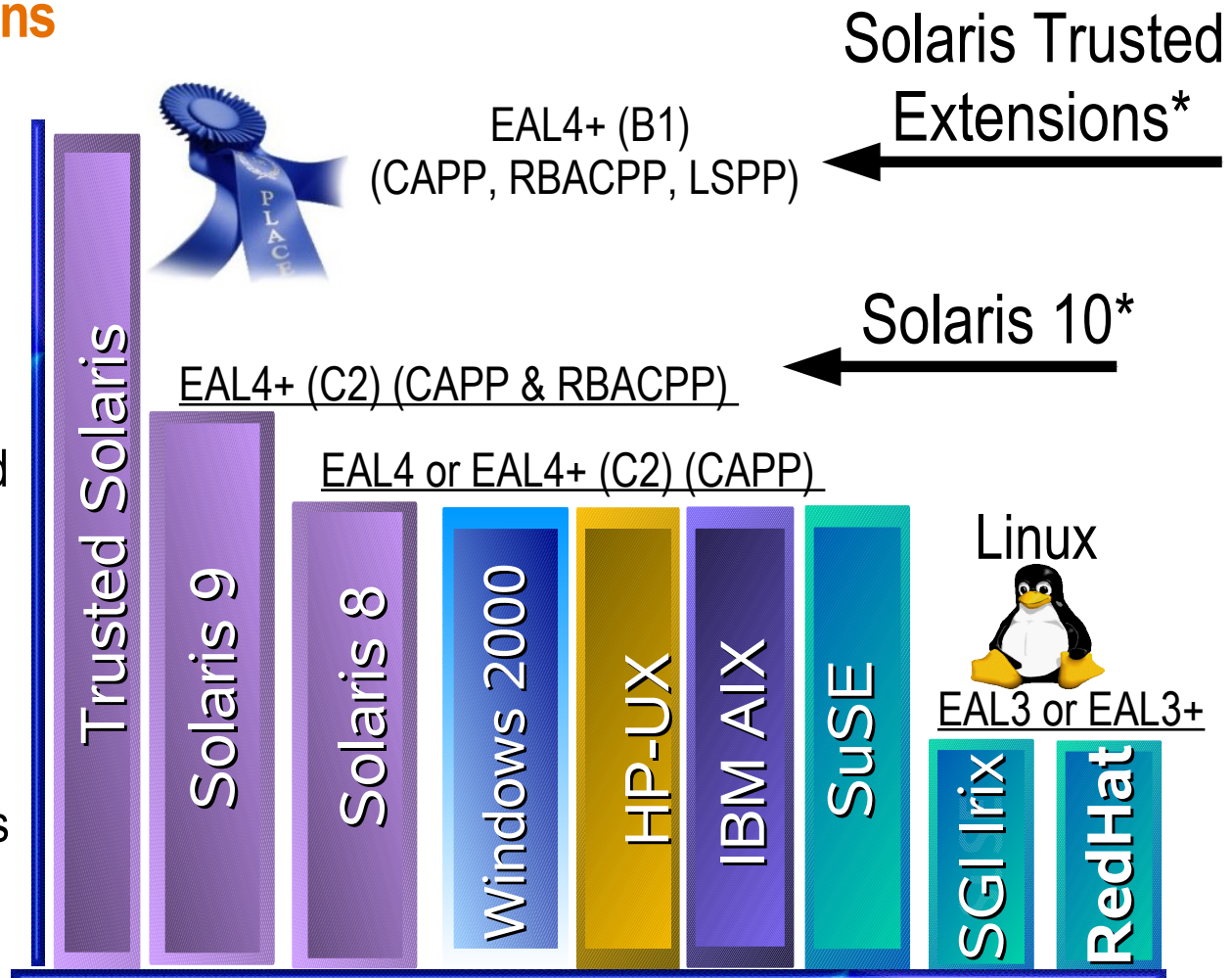
Available Q3CY2006

*Single-Level Java DS in first release; Multi-level Java DS @ release or later

Solaris Security: Independent Validation

3rd Party Certifications

- Trusted Solaris 8:
Only general purpose OS certified with 3 Protection Profiles at EAL4+
- Solaris 9 is evaluated at EAL4+ with CAPP and RBACPP
- High degree of commitment to independent certification of Solaris



Based on data from <http://www.commoncriteriaportal.org/>

* Future evaluations

Sun Proprietary/Confidential: Internal Use Only

Solaris 10: A Generation Ahead



Extreme Performance



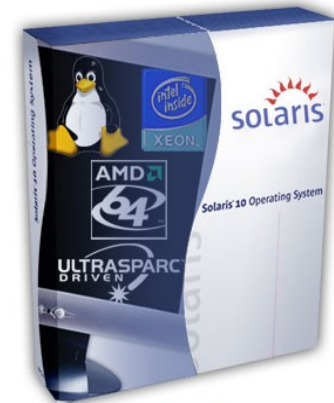
Optimal Utilization



Unparalleled Security



Relentless Availability



Platform Choice

Linux Interoperability

- **Linux** applications
- Solaris performance, scalability and security
- Reduced deployment costs
- Freedom to choose applications



Platform Choice

Linux Compatibility

Solaris Containers for Linux Applications

- Lets current Linux applications take advantage of Solaris performance, scalability and security—without recompiling
- Next-generation *lxrun* replacement
 - > *lxrun* = user-space emulation layer
- Direct support of Linux system calls



Platform Choice

Linux Compatibility

Solaris Containers for Linux Applications

- Run unmodified Red Hat applications
- Extension of Solaris 10 Container technology
- Designed to support different “brands”
- Leverages all Solaris security / administrative advantages
- Combines the best of virtualization, resource management and OS flexibility

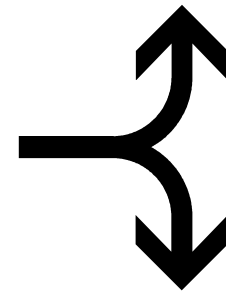


Platform Choice

Key Features and Benefits

Deployment Sweet Spots

- Co-existence of Solaris and Linux environments
 - > Keep Linux-only applications
 - > Test and deploy in different environments
 - > Fine-grained, multi-tier administration and user rights
- Leverage OpenSolaris for additional Brands



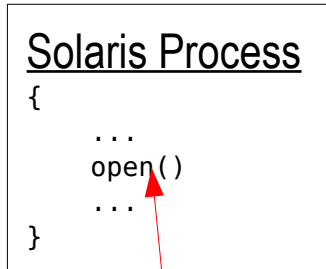
Key Features and Benefits

Deployment Sweet Spots

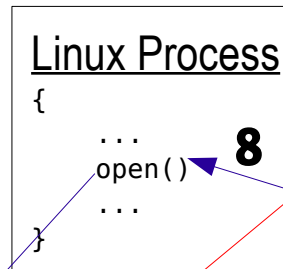
- Cross-Platform Development
 - > Develop Linux applications on Solaris system
 - > Ideal for ISV / Developer
- Observability
 - > Code, debug and tune Linux applications on Solaris
 - > DTrace – Monitor complex, heterogeneous n-tier system on single machine
 - > mdb, gdb, strace
- Goal: Be a better Linux development platform than Linux

Architecture Overview

Solaris Container



Linux Container



LX emulation library

```

lx_open(args) {
    fd = open((lx_to_solaris(args))
    if (fd < 0)
        return (solaris_to_lx(errno))
    else
        return (fd);
}

```

Userland

A,D

1

8

Kernel

4,7

3

Solaris Kernel

Syscall handler

```

if (p->p_brand)
    p->p_brand->br_syscall();
else
    rval = do_syscall();
return to userspace

```

2

**5,6
B,C**

```

open() {
    ...
    return(fd);
}

```

LX brand module

```

struct lx_brand_ops {
    lx_syscall()
    lx_proc_exit()
    lx_pid_assign()
    lx_pid_release()
    lx_setregs()
    ...
}

```

```

lx_syscall {
    return to userland
}

```

BrandZ – Zones Integration

- A Brand is an attribute of a zone, set at zone create time
 - > `zonecfg create -B <brand type>`
- Virtual platform is defined in per-brand XML file
 - > Per-brand control over filesystems and devices
- Each Brand provides install and boot scripts
 - > Live in `/usr/lib/brands/<brand>/`
 - > Install script loads software into the zone
 - > pre-boot/post-boot scripts allow the brand to create device nodes, perform any configuration, etc.
- `'zoneadm list'` will report brand types

BrandZ – Kernel Support

- BrandZ infrastructure provides a set of interposition points:
 - > syscall path, process loading, signal delivery, etc.
- Interpositioning allows a Brand to replace Solaris behavior with its own
- Only applied to processes in a branded zone
- Fundamentally different brands may require new interposition points

The 'Ix' Brand

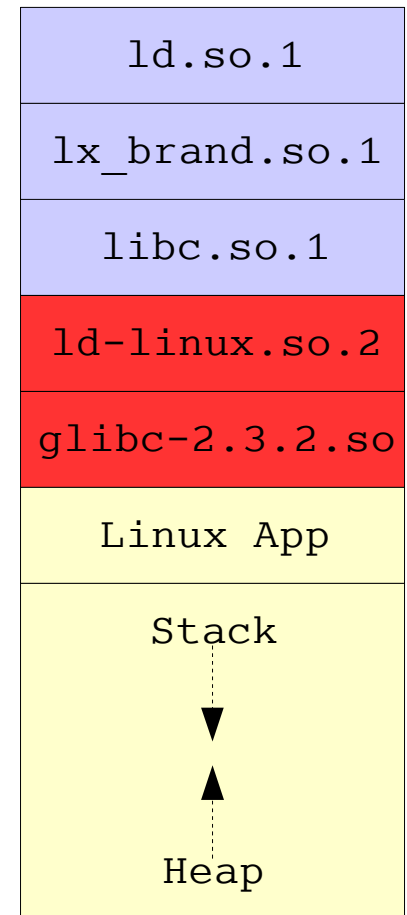
- SCLA is implemented as a Brand
 - > Not just a plug-in brand; required a new interrupt handler.
- Install script populates a zone with Linux bits
 - > Customer must provide RHAS or CentOS install media
 - > Will also support flasharchive-like installation from a tarball
- Most Linux emulation is done in user space
- Linux /proc implemented as an in-kernel filesystem

Launching Linux Processes

- Linux and Solaris both use the ELF executable format
 - > Unmodified Solaris can parse and load a Linux binary
- In branded zones `elfexec()` calls brand-specific `exec()` routine
 - > Loads our linker and library as well as the Linux linker and binary.
- Added 7 `AT_SUN_BRAND_*` fields to the aux vector
 - > Hold information about Linux linker and binary
 - > Used to construct an aux vector for the Linux linker

Running Linux Processes

1. Kernel jumps into our linker
2. Loads our `libc` (and a few others)
3. Resolves symbols in our libraries
4. Runs `_init()` in `lx_brand.so.1`
 - Pass `lx_handler()` address to kernel
5. Builds aux vector for Linux app
6. Jumps to Linux linker
7. Load Linux `glibc`, etc.
8. Resolves Linux symbols
9. Jumps to Linux `main()`



Address Space Map

System Call Emulation

- Linux uses interrupt 0x80 to enter kernel
 - > Solaris uses syscall, sysenter, lcall, or interrupt 0x91
 - > Required adding a new interrupt handler
 - > Demonstrates challenges faced by third-party brands
- We immediately jump to user-space brand library
- Brand library generally calls Solaris equivalent
 - > Possibly with some argument or rval mangling
- Complex system calls are implemented in 1x kernel module
 - > Use new `SYS_brand` system call to enter kernel module
- When done, we return directly into application

/proc

- Implemented as a new file system
- Linux /proc is a Superfund site
 - > Starts getting cleaned up in 2.6 kernel
- We will support a subset of its functionality
 - > /proc/<PID>/*
 - > /proc/loadavg
 - > /proc/meminfo
 - > /proc/mount
 - > /proc/stat
 - > /proc/uptime
- Most process info maps directly to our /proc
 - > Mount Solaris /proc in the zone under /native/proc

Observability

- Support both Solaris and Linux tools
- In the Linux zone:
 - > strace – syscall tracer
 - > gdb – GNU debugger
- From the Global zone:
 - > Dtrace: will have PID provider and Linux syscall provider
 - > mdb: will be able to manipulate live processes and core files
- Goal: to be a better Linux development platform than Linux

System Choice

Optimized Solaris Performance on a Variety of Systems

- AMD Opteron
- CMT enhancements
- Pentium 4 SSE
- Hyperthreading



Platform Choice

Software Express for Solaris

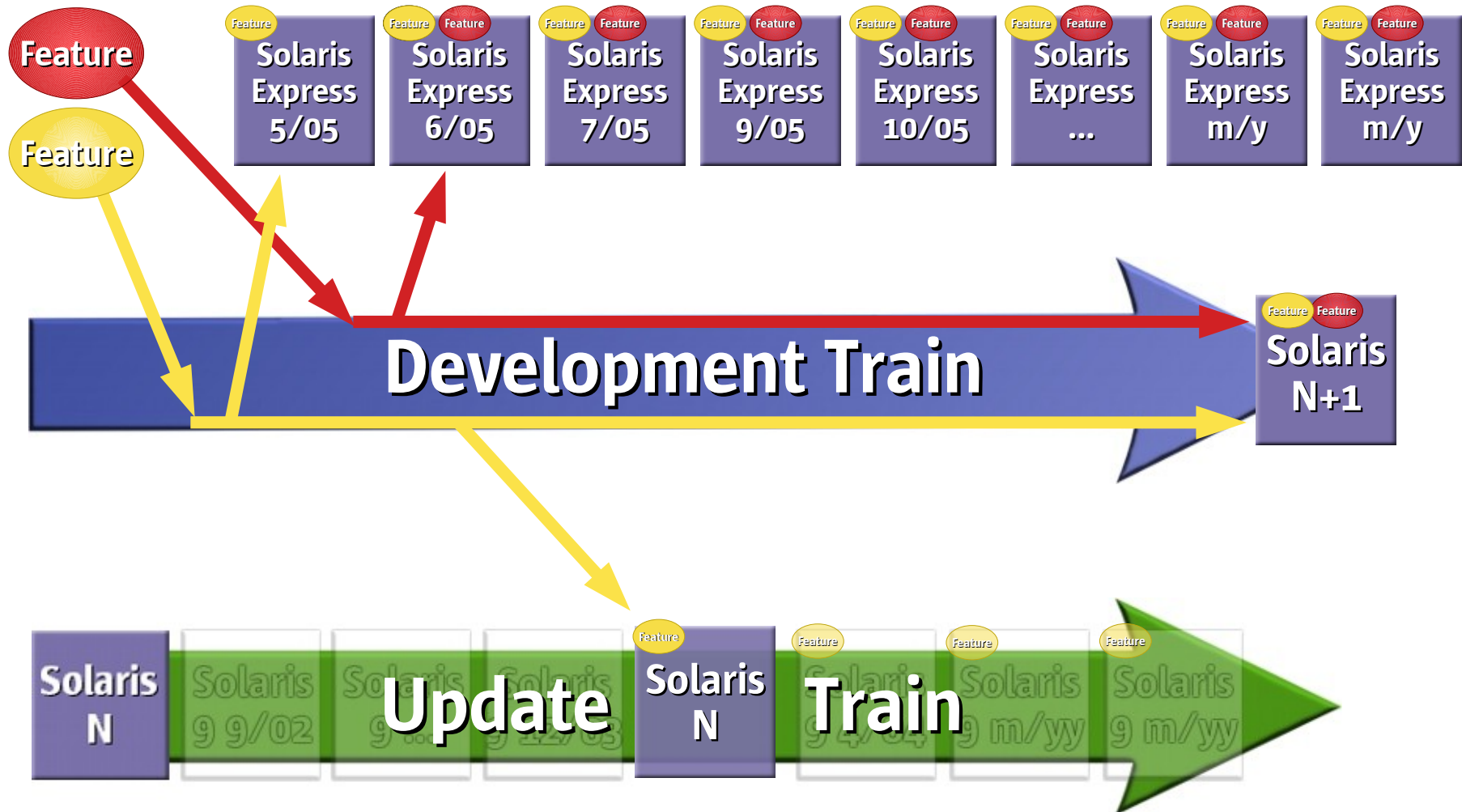
Timely Access to Leading-Edge Technology

- Monthly builds of next Solaris release
- Early access to new features
- Free access for downloads
- Subscription support for \$99/year

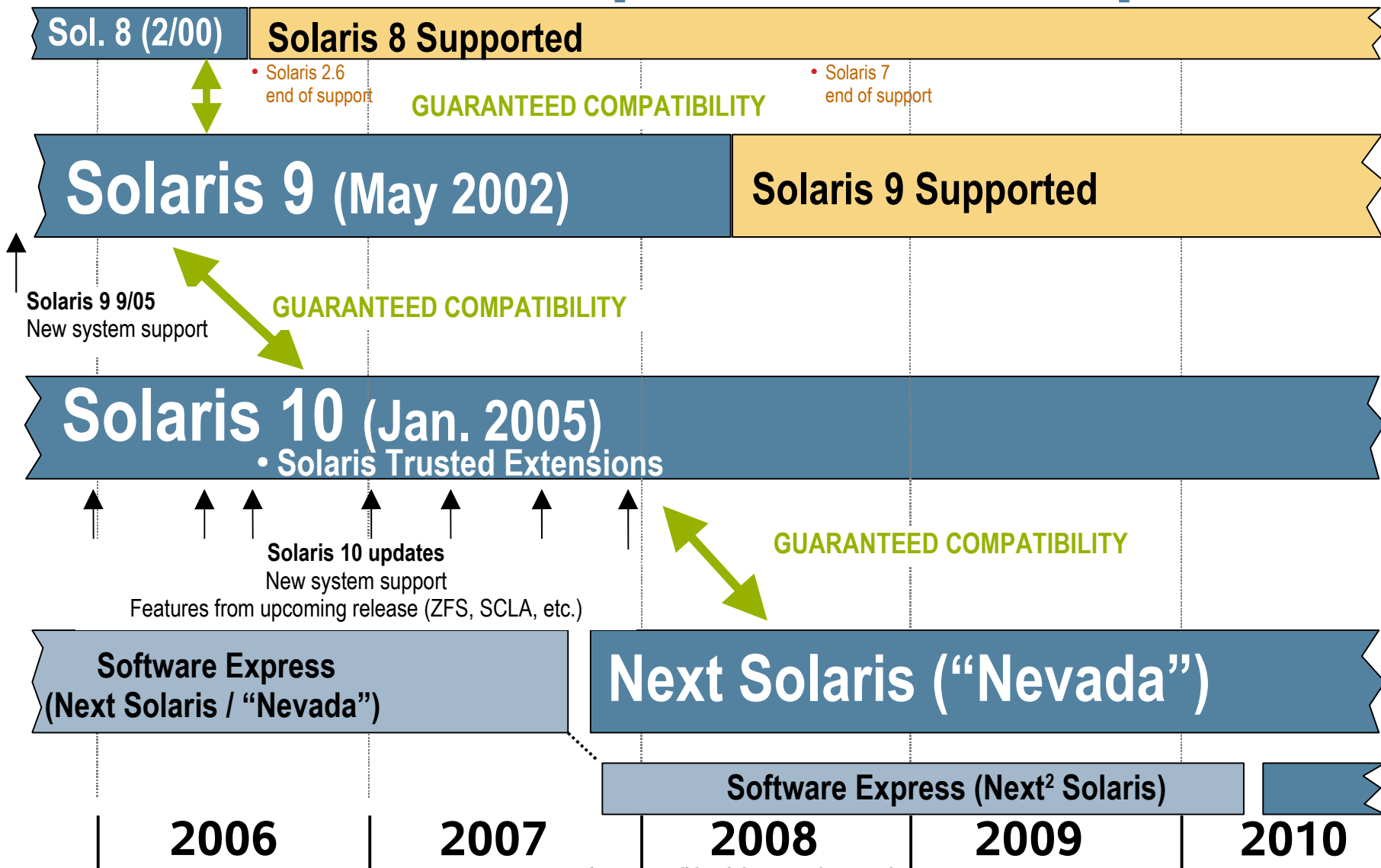
Accelerated delivery of disruptive technology

Software Express for Solaris

Timely Access to Leading-Edge Technology



Solaris Roadmap, Oct '05 – Sep '10



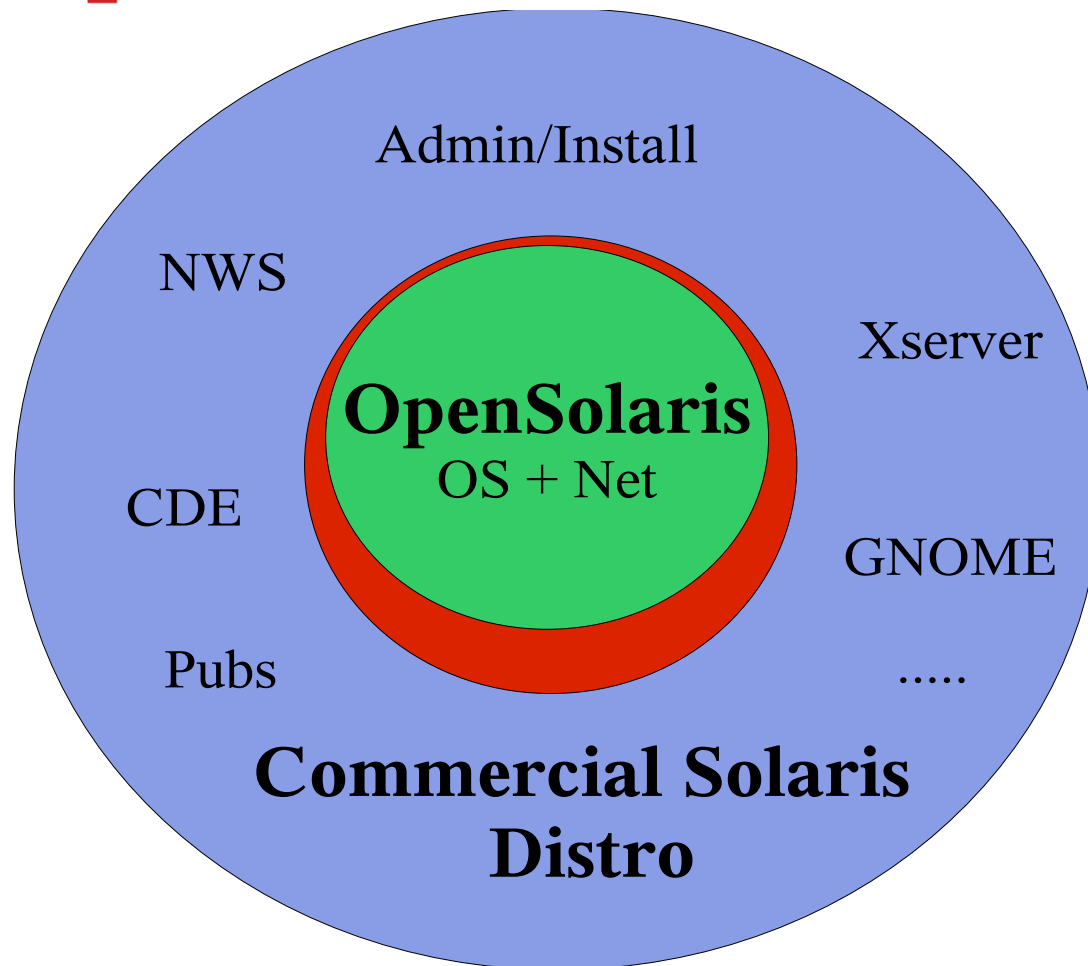


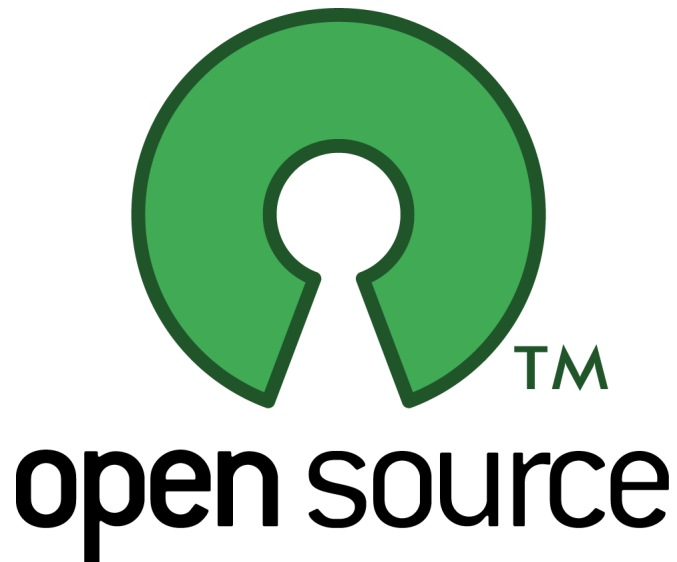
\$0

opensolaris™

A light blue reflection of the word "opensolaris" is positioned directly below the main text.

opensolaris™





OpenSolaris

Solaris Source Code

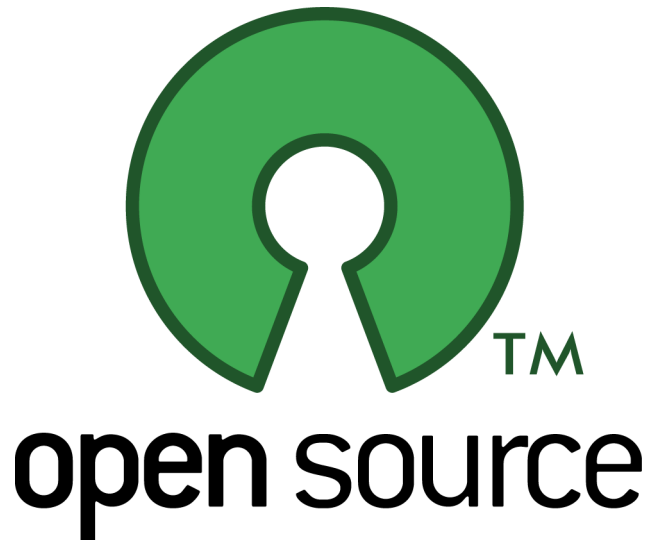
OSI Approved License

Patent Commons

Buildable Source Q2CY2005

www.opensolaris.org

CDDL



Based on Mozilla MPL

OSI approved

Royalty free

Modifications must be shared

Can mix CDDL source files with
files covered by other licenses

Blanket patent grant

Patent protections



opensolaris™

Sun design principles

Sun's high-quality standards

Transparency

OSI-approved licensing

Sun-sponsored

A Different Approach

- Open
- Investment protection
- Performance improvements on every architecture
- Consolidation without complication
- Self-managing systems
- Rapid feature innovation



Open Source is Mainstream



Participation Age Implications



- Ubiquitous networks
- Internet utilization
- Standardized platforms
- Shared technology/ideas
- Lowered barriers to entry
- Community software
- Near-universal access

Open Source in a Nutshell

- A community of developers
- Sharing a code commons
- Creating “wealth” from the commons
- Enriching the commons in the process



The “craft guilds” rediscovered...

... *Connected* Capitalism

“ Open source is
about **community**,
not economy
not hype,
not Linux
not OpenSolaris
Not...”

Open Source Is Here to Stay

Ask Your Peers

Internet



Fortune 500



Government



Ask Your Employees

IT Departments



At Home



Moonlight Programmers



Why Sun and Open Source?

- It's the right solution for our customers
- It opens new opportunities for Sun
- It's the right business model for NOW!



What is Open Source?

Using Shared Resources to Solve Needs and Create Wealth

- Distribute binaries and source code
- Freely modifiable and re-distributable
- Non-discriminatory
- Consensus driven projects
- Meritocracy
- Peer review and public discussion
- OK to make money - but not for access to code

Why Open Source?

- Massive peer review means higher overall quality
- Promotes unexpected innovations
- Creates opportunity for developers to innovate and develop new applications on the shoulders of existing ones
- In the “Participation Age” open source communities share technology & knowledge to refine and improve their own businesses, communities and society

Benefits of Open Source

Good For Customers

- Community drives competition, choice, value
- Accelerates unexpected, disruptive innovation

Good For Sun & Partners

- Innovation happens everywhere
- Creates new opportunities by growing the market

Myths About Open Source

1. Open Source developers will build your project for Free
2. Open Source is anti-business
3. Open Source projects are hostile to corporate developers
4. Open Source software never really ships on time (or at acceptable quality)
5. Nobody really makes money

Truths About Open Source

1. 45% of CIOs in the Fortune 2000 are looking for an alternative to proprietary OS
2. Linux: massive worldwide deployments
3. Open Source no longer synonymous with “free” or no/low value technology
4. 55% of Open Source developers at OSDN are paid to work on at least one Open Source project
5. Venture capitalists are investing in open source companies
6. Companies can redistribute open source software and charge for different value-added services

Open Source Monetization

- Two Golden Rules:



- > ***Collaborate***
over what does not differentiate
- > ***Compete***
by innovating on the commodity
base

Open Source is OPEN if:

- License does not restrict:
 - > **Parties** to whom the code may be licensed
 - > **Uses** to which the code may be put
 - > **Software** of which it may become a part
 - > **Pricing** of the software using the licensed code
- Source easily and inexpensively available
- Modification to the source permitted at least to the degree patches are permitted
- License must not affect licenses of other code distributed with the licensed software

The Original Idea



- **Sun executives** – expand the Solaris market, drive Solaris into new markets, sell more systems & services.
- **Solaris engineers** – engage developers outside the company, share code, improve an already great system.
- **Solaris market/community** – see the code, optimize apps, contribute to Solaris development, create ports/distros.

10 M Lines of Source Code

Core Operating System

kernel, rctl_action, dtrace_probe
pool_bind_kmem, create, create_enter, chipstrutx_exit,
ddi_fm_capable, priv_set_t, putnext, lgrp_mem_rename,
syscall_mstate, vmem_xfree

Networking

tcp, dhcp, ipsec, nfs, dlp, dns, ldap, nis, nis+, ppp, ipqos, ip
multicast, ip_multipathing, ipv4, ipv6, ipc, udp, snmp, sctp,
packet filtering

System Libraries

libc, libumem, libsysevent, librt, libnsd, libproc, libsocket, libscf,
libw, libkstat, librpcsvc, libxnet, libcourses, libbpm, libnvpair,
libsendfile, libadm

Commands

Kstat, ifconfig, zoneadm, svcadm, traceroute, ppriv, prctl, mdb,
pfexec, lofiadm, lari, ifconfig, ifadm, dispadmin, cputrack, crle,
ptree

The OpenSolaris Community

Browse

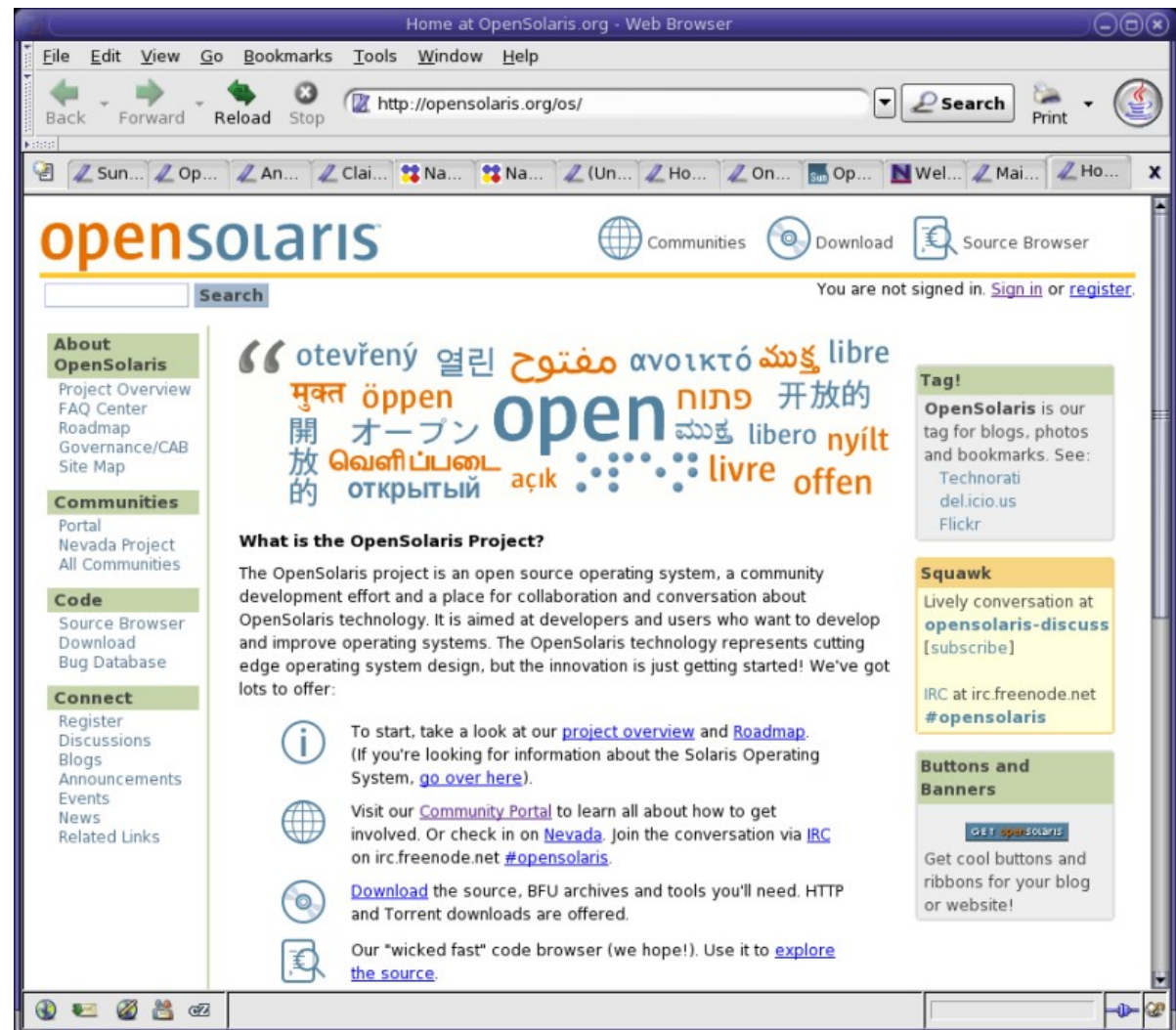
Source code
FAQs
Communities
Discussions

Download

Source Code
Binaries
Tools

Participate

IRC #opensolaris
Discussions
Blogs
Communities
User Groups
Buttons



Solaris Business Model 2004

Services:

- Education
- Support
- Consulting
- ...

Sun Services

3rd Party

Applications:

Operating Sys:

Solaris

Hardware:

Sun Hardware

Other Hardware

Solaris Business Model Now...

Services:

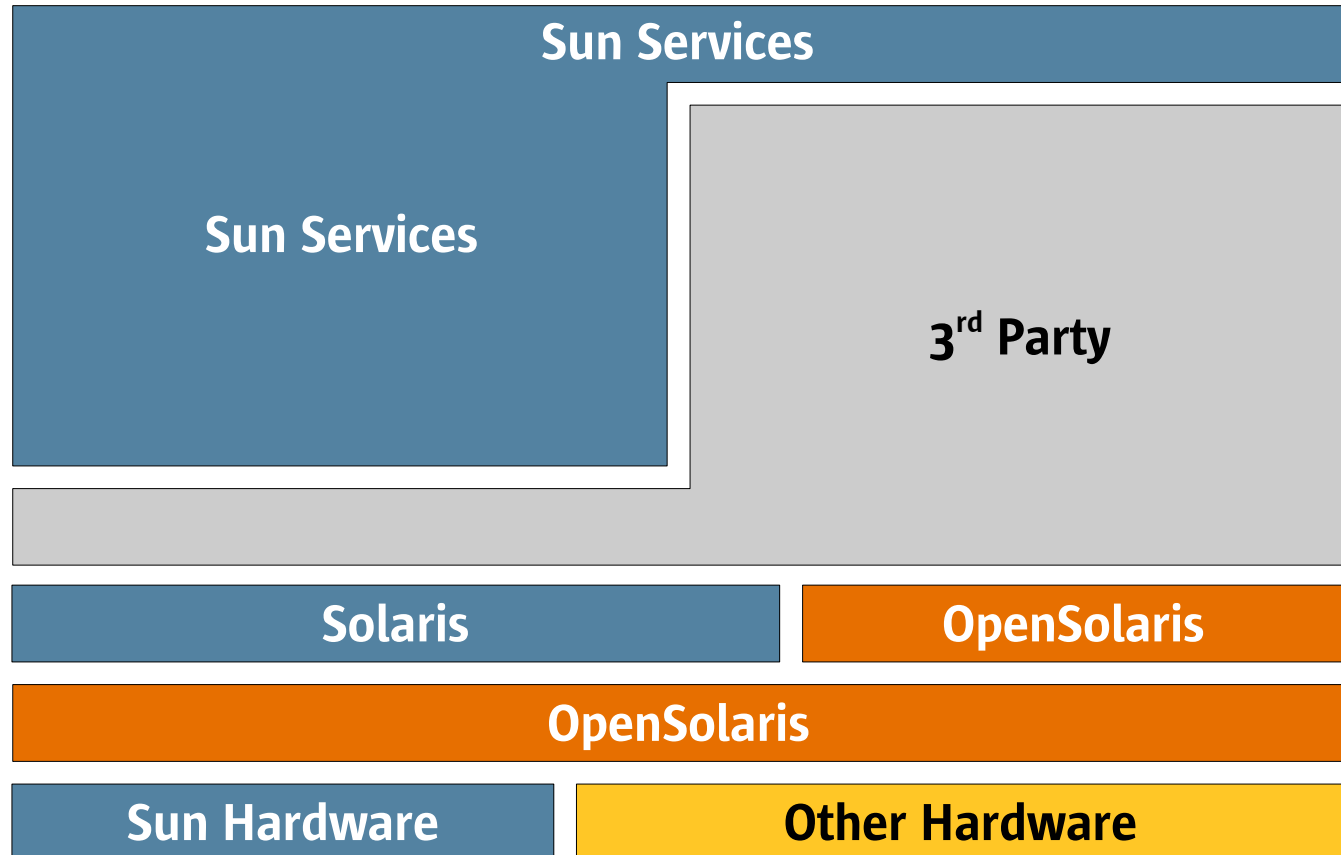
- Education
- Support
- Consulting
- ...

Applications:

Operating Sys:

Community:

Hardware:



Sun and Open Source

Sun Initiated and Sponsored



Open Source



Standards and Industry Groups



Top 10 Things to Know

1. Sun has open sourced the Solaris Operating System
2. Solaris is now an open source operating system
3. OpenSolaris IS an open source project, a source base, a community, a website – NOT a distribution
4. Next release of Solaris will be built from the OpenSolaris code base
5. OpenSolaris contains source for all the major innovations in Solaris 10 – things like DTrace, Containers, Predictive Self Healing...
6. All the development tools you need – including free Studio compilers for community members, “wicked fast” code browser, bite-size bugs...
7. True open source – CDDL is OSI-approved, best license for businesses
8. OpenSolaris is a next-generation platform for innovation
9. OpenSolaris community is growing and active – see the new community live-CD distributions based on OpenSolaris, see the FreeBSD port of DTrace, see the putbacks
10. OpenSolaris grows the ecosystem for the Solaris technology – which floats all of our boats

License Comparison

Three OSI Approved Licenses	CDDL (Common Development and Distribution License)	GPL (GNU General Public License)	MPL (Mozilla Public License)
Patent Protection	Most	None	Some
License includes an express patent grant	Yes	No	Yes
License taints larger work	No	Yes	No
Can distribute binaries under your own license	Yes	No	Yes
License compatible with other copyright licenses	Yes	No	Yes
License widely adopted	Not yet	Very	No
Modifications must be published	Yes	Yes	Yes
Can be integrated with proprietary software	Yes	No	Yes

Opening the Door to Innovation

- Sun's OpenSPARC initiative intends to open source UltraSPARC T1 design point in 2006
 - > Target date: March 2006
 - > Details to follow in Q1CY2006
- Initial publications should include:
 - > Source of the design expressed in Verilog
 - > A verification suite and simulation models
 - > Instruction Set Architecture (ISA) specification (UltraSPARC Architecture 2005)
 - > A Solaris port



Sun is the only vendor to open source its 64-bit processor design

What Open Sourcing the Chip Enables

Opening the Chip Design Creates Opportunities for Innovation...

Market Advantages

- Seeds a new SPARC eco-system
- Speeds innovation into the market
- Supports the growth of a thread-rich application environment

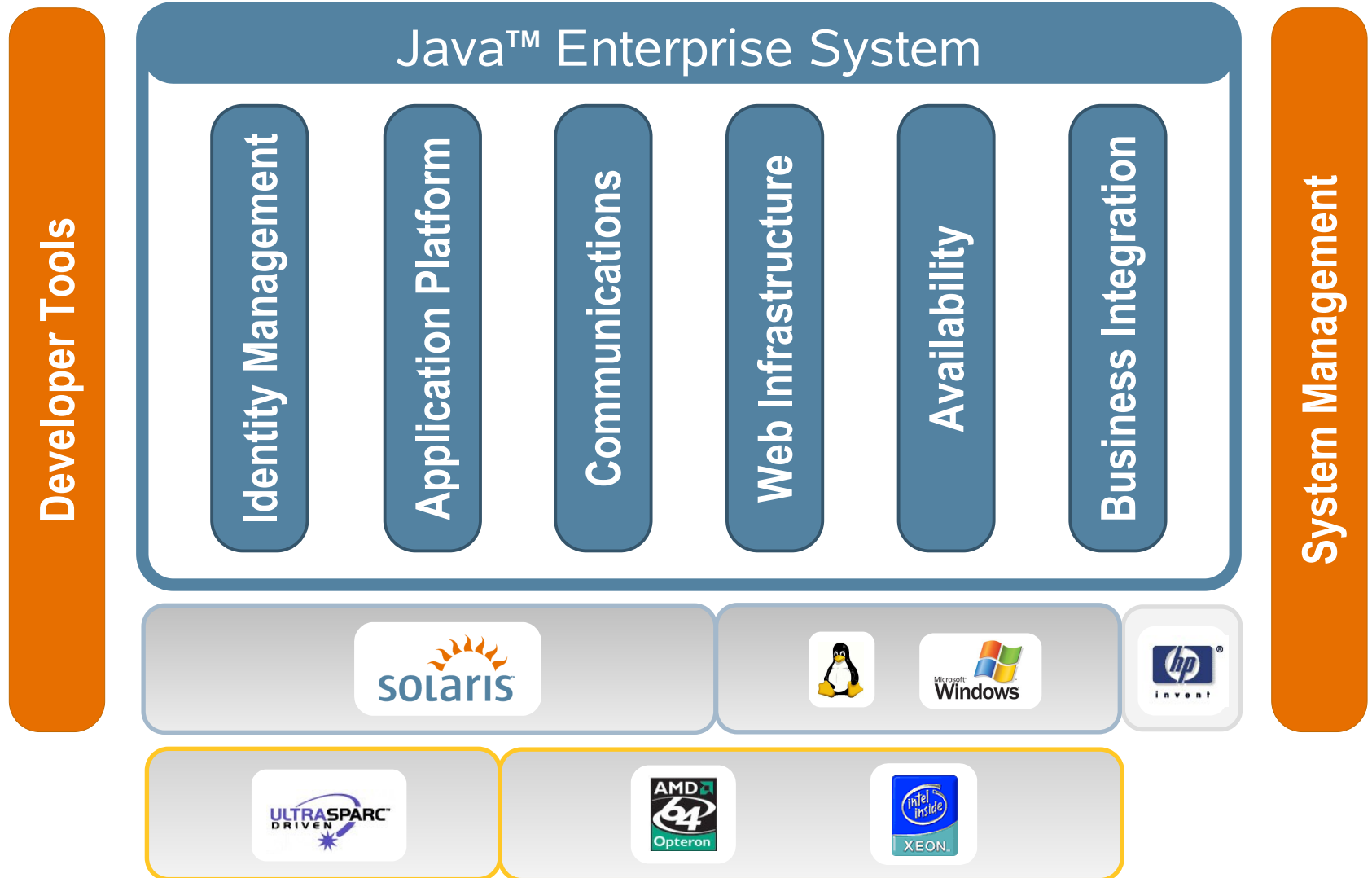


Developer Advantages

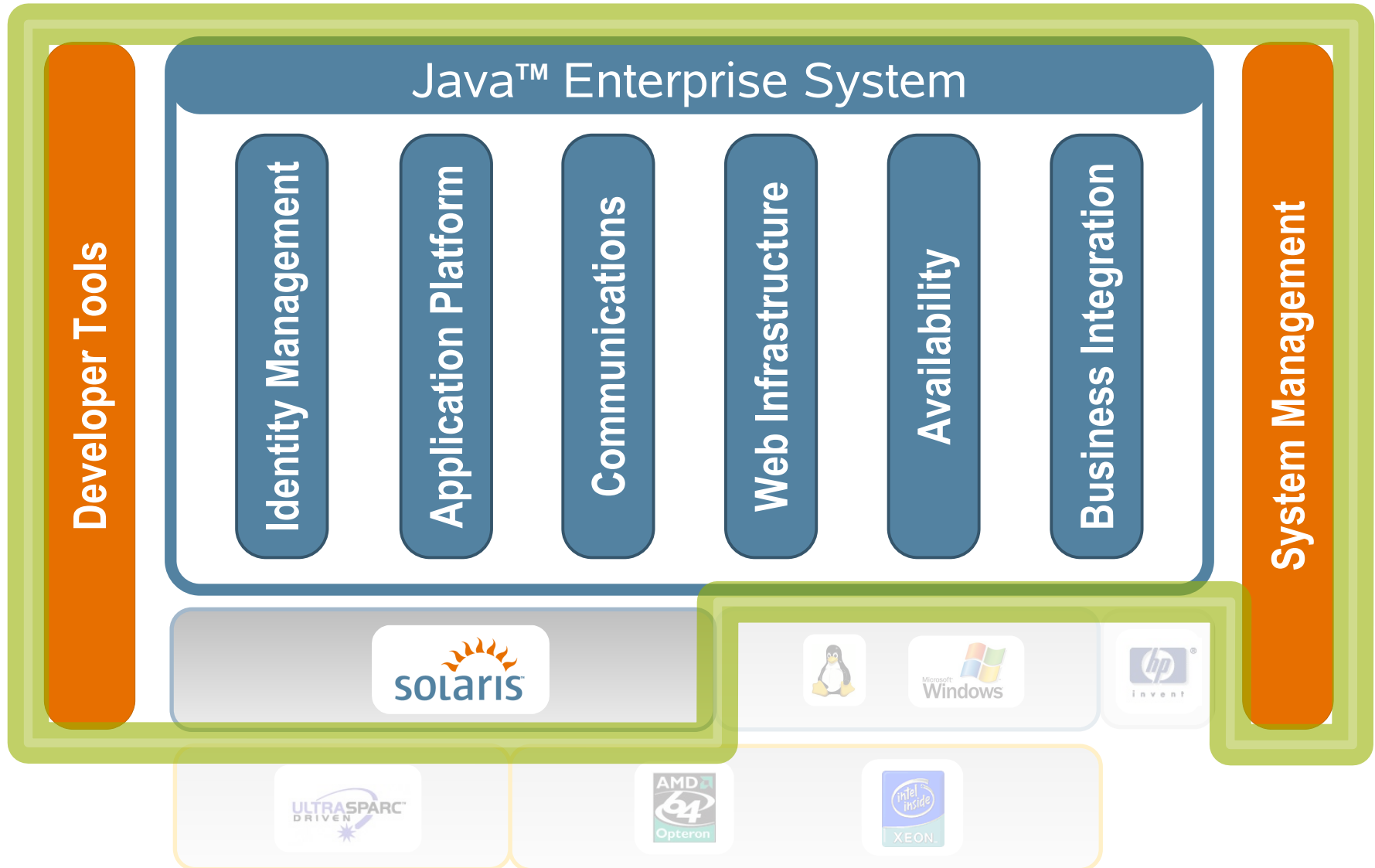
- Dramatically lowers the cost of entry
- High-performance, 64-bit design
- Access to a 32-thread, modern architecture
- It's ***FREE!***

Sun will be the *only* vendor to make its processor design available to the open source community

Sun's Software Portfolio



Solaris™ Enterprise System



SOLARIS

Carlos Piedrafita

carlos.piedrafita@sun.com