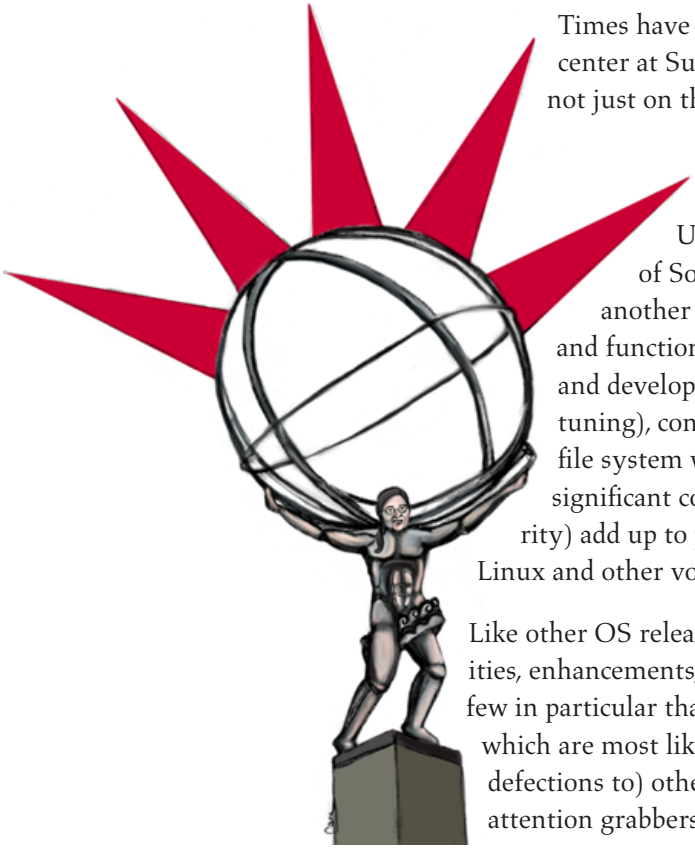
Solaris Rises

Research Note

Gordon Haff
12 November 2004

Solaris, *née* SunOS,[1] was once a major part of what defined Sun Microsystems as a company. It was the flavor of the Unix OS that developers most favored. That preference, in turn, opened innumerable datacenter doors for Sun's sales force. But, over time, Sun's focus on Solaris blurred. It was no longer the main attraction; it became a supporting foundation in a world that increasingly wrote to high-level languages and APIs (e.g. Java and J2EE) rather than the operating system or "bare metal."[2] As it cut costs and cleaned house, Sun even dropped Solaris' x86 version in favor of Linux, bowing to the upstart in light of the Linux enthusiasm among developers and the lackluster ISV support of Solaris x86.



Times have changed. Solaris 10 is once again front-and-center at Sun—as much as, or more than, it's ever been. And not just on the company's beloved SPARC processor architecture. For the first time ever, Solaris x86 is in the mainstream.

Underpinning this shift in strategy is the rollout of Solaris 10, a significant new release. More than yet another OS release, Solaris 10 introduces new features and functions that are directly visible and relevant to users and developers. Features such as Dtrace (performance tuning), containers (isolated resource partitions), ZFS (a new file system with integrated volume management), and significant convergence with Trusted Solaris (enhanced security) add up to product that's significantly differentiated from Linux and other volume OSs.

Like other OS releases, Solaris 10 is a large grab-bag of new capabilities, enhancements, and fixes—over 400 by Sun's count. But it's a few in particular that make Solaris 10 stand out from the pack, and which are most likely to attract converts from (or stem defections to) other platforms. Let's take a closer look at these attention grabbers.[3]

1. Technically, SunOS lives on as the core operating system component of the entire Solaris environment. In the case of Solaris 10, the core OS is SunOS 5.10. But this is a largely pedantic distinction; essentially everyone just calls the product Solaris.
2. See Illuminata report "Solaris 9: Fortifying the Foundation" (July 2002).
3. See the table for a broader list of Solaris 10 features and their significance.

Major New Solaris 10 Features

(Those in italics receive in-depth coverage in this report)

Solaris 10 Feature	Description	Comment
Solaris Containers	Namespace-isolated workload groups that present the appearance of independent OS instances with low overhead.	Provides less isolation than traditional partitioning approaches, but strong example of increasingly accepted, lightweight style of workload separation.
DTrace	Flexible and extensive probes infrastructure for observing and tuning system performance.	Early customer experiences extremely positive; concept not unique to Sun but market-leading implementation suitable for use on production systems.
ZFS	New 128-bit local file system that integrates volume management functions.	Little field experience yet. Capacity increases interesting but management simplification and performance increases of greater near-term interest.
Process Rights Management	Breaks root capabilities into a granular set of privileges, allowing processes to be granted only as much power as they need to do their assigned task.	The latest in a series of Trusted Solaris concepts and technologies to be rolled into the mainstream OS.
Reworked TCP/IP stack ("FireEngine")	A new packet classification engine shunts packets off to special routines that can deal more effectively with specific packet types such as UDP, TCP or IP.	Creates a framework for future offload to specialized offload engines or a dedicated core on a multi-core chip.
Solaris Cryptographic Framework	Optimized implementations of common cryptographic (encryption and message hashing) algorithms and a plug-in framework to abstract the implementation.	Reduces the need for separate and independent crypto implementations. A simple change of policy can switch from one type of crypto to another.
Project "Janus"	Lets the Solaris kernel identify Linux applications and service Linux systems calls. This lets Linux binaries run on Solaris as if they were native Solaris applications.	Good coexistence/migration element, but users must install the relevant Linux packages of supported Linux distributions (e.g. libraries, services) that are needed to run Linux applications under Janus.
Solaris Fault Management ("FMA")	Allows Solaris to diagnose problems automatically, potentially predict them, and react in a fine-grained way. Correlates error messages to failed components and required or recommended corrective action.	Framework and monitoring agents largely in place; further diagnosis code (e.g. for AMD64 in Update 1 in 2005) still to be developed. Major component of "Predictive Self Healing" along with Solaris Service Manager.
Solaris Service Manager "Greenline")	Allows only the software components affected by a hardware or software fault to be restarted, rather than the entire system or partition.	XML files that describe dependencies still largely to be developed by ISVs and users. Initially manual restart; more automation over time.
NFS v4	Integrates locking and mount and bundles multiple RPCs together in a single exchange; adds RPCSEC_GSS API.	Improves WAN performance and security.

A New Way to Slice and Dice

Of all the new features in Solaris 10, the company has trumpeted "Solaris Containers" the loudest and the longest. For a full OS generation, in fact; Sun was already discussing containers as a future while previewing Solaris 9. Originally codenamed "Kevlar," Sun referred to them until recently as "N1 Grid Containers" but has now dropped that awkward and inappropriate term.[4]

Different IT vendors apply the "containers" tag to somewhat different things. Trigence, for example, uses the term for the application wrappers it uses to simplify deployment and failover. J2EE application servers use the term to describe the protected collections of classes and objects typically deployed for a given application. But Solaris Containers build off the workload group concept that resource managers like AIX's WLM, HP-UX's PRM, Solaris's S9RM, and Windows WSRM use to corral multiple processes and control their resource use as a single entity. Solaris Containers then go a step further by "hardening the walls" between these groups and creating an illusion of independent systems.[5] For example, a Solaris Container can have its own IP address just like a regular full system can. BSD jails are a progenitor of Solaris Containers. SWsoft's Virtuozzo is a related contemporary counterpart.[6] HP-UX Secure Resource Partitions is an upcoming one.

Containers provide isolation that is intermediate between partitions—including virtual machines (VMs) and traditional workload groups. Like partitions and VMs, a container presents the appearance of being a separate and independent OS image—a full system, really. But, as with the workload groups that they extend, there's only one actual copy of an operating system running on a physical server. Containers are fundamentally a lighter-weight approach than partitions or VMs; all of the container instances on a system share a great deal of the underlying OS code. By contrast, each partition or VM requires a full OS copy. Thus, it's often possible to have far more containers than VMs on a given system. Sun touts a containers-per-system maximum of over 8,000. Tens to hundreds will likely be more realistic limits, depending on the size of system and how many resources the applications need.

The flip side is that containers isolate less than more heavyweight approaches do. Containers may prevent application failures from affecting other applications or application instances on a system. They also hide resources from programs running in other containers, improving security.[7] But containers offer no protection against a failure of the underlying OS, of which there is but one copy. To be sure, having a single OS image also has advantages. There's only one copy to update and patch, for example. Nevertheless, it does create a single point of failure and a potential performance bottleneck, making other Solaris availability and performance enhancements that much more important.

Sun's enthusiastic adoption of the container concept is a *radical* about-face from a few years back. Sun's long-held partitioning story was that electrically-isolated physical partitions (a.k.a. Dynamic System Domains) were the only way to fly; anything less fault-isolating was *'way* too risky.[8] However, as individual processors have increased in performance, and as small systems began playing a larger

4. Understandably—given how Sun has de-emphasized both N1 and grid computing while re-emphasizing Solaris. Besides, Containers never had much to do with either N1 or grid anyway. Sun's also used the term "zones" in the past; although it lives on as an underlying technology and administrative concept, Sun has banished zones from its marketing vocabulary.
5. By analogy to programming language "namespaces," Containers are zones within which names must be unique but which provide a unique context relative to other namespaces. In a sense, containers are "namespace-isolated resource groups."
6. See Illuminata report "Virtuozzo: The Lighter Side of Virtual Machines" (August 2004).
7. It's hard to attack what you can't see or control.
8. Sun consistently attacked HP's PRM and vPars, IBM's LPARs (even on the mainframe!), and VMware virtual machines as insufficiently robust partitioning approaches even though they, in fact, offer *more* isolation than do containers.

role in datacenters, course-grained physical partitions[9] are increasingly a relatively niche technology for heavyweight applications running on Big Iron. Important in their own way, but far removed from the volume systems mainstream where more software-based approaches from both system vendors and ISVs like VMware hold sway. Containers give Sun the software-based response that it has sorely lacked. This is key, as small, powerful, inexpensive systems based on Opteron assume an ever more important role at Sun.

Helping Tune Performance

Sun hasn't been talking about DTrace quite as long as it has about containers, but based on the reactions of some early access customers, it's almost as impressive a feature of the new release.

Performance tuning is *hard*. It's hard even just to find and identify the serious problems among all the possible issues in all the possible places. Existing trace tools tend to have a fairly limited number of predefined trace points. For example, Opersys' Linux Trace Toolkit has a total of 48—and requires a custom Linux kernel so that the events can be logged.[10] Profilers and debuggers often provide more flexible options, but they enormously slow the system they're observing. As a result, they're not only unsuitable for production use, but have to be used selectively even in test and development. Casting a wide net and constantly monitoring a large number of potential variables just takes too long and collects too many mountains of data. As a result, monitoring and debugging tools are most effective at pinpointing the source of performance issues whose general location and cause are already somewhat understood.

DTrace stands for "Dynamic Tracing" and, indeed, it's that dynamism that most distinguishes it from other approaches. A developer, administrator, or performance tuner uses the DTrace scripting language to dynamically establish monitoring points of interest, whether in the OS kernel or user processes. A probe is a location or activity to which DTrace can bind a request to perform a set of actions, like recording a stack trace, a timestamp, or a function argument. Think of them as programmable software sensors that gather interesting information about the system and report it. DTrace in Solaris 10 comes with something like 37,000 predefined probes; users can also define their own. Probes come from a variety of kernel modules that Sun calls *providers*, each of which creates a unique type of instrumentation. For example, there's a provider that creates a simple time-based counter (`profile`) and another to understand lock contention and other sorts of locking behavior (`lockstat`). But essentially *any* function entry or exit is a potential probe location; DTrace hot patches the function entry point in memory to insert the probe.

However, what's most striking about DTrace isn't its technical details, but how enthusiastically it's been embraced by many Software Express downloaders.[11] Sun has obviously carefully selected the examples it has trotted out, but they're no less impressive for that. For example, one company increased the performance of its financial database application 32 percent within just a few hours of trying out DTrace—and this was an application they had been tuning for years, gaining just a few percentage points of performance here and there. Sun says that they have seen up to a tripling of performance, with doubling not uncommon.[12]

9. Hardware partitions typically split at the electrical connections between building blocks or system boards. This technique therefore only applies to relatively large systems; given that most such blocks contain four processors, eight-way is the entry point for this partitioning style, and "interesting" configurations take upwards of 16 or 20 CPUs.

10. Some commercial Unix products such as AIX have more trace points (hooks in IBM parlance, about 400); both the hooks and their associated actions are statically defined. IBM also has a Dynamic Probes debugging facility for Linux that has some of the dynamic characteristics of DTrace.

11. A widespread, open beta program that Sun is using to spread advance copies of Solaris 10.

12. Such "we made fabulous gains in an account that can't be named" examples are of course inherently suspicious. But our discussions with performance consultants using DTrace, as well as interactively driving the monitoring process ourselves, are both quite compelling.

These are not trivial performance increases. Yet the work it takes to identify the problems is, if not trivial, certainly small indeed relative to the result. If these early DTrace anecdotes and experiences are indicative of what a broad set of customers will achieve, DTrace alone would be reason enough for many users to get themselves on Solaris 10 sooner rather than later.[13]

ZFS

Solaris 10 also debuts a new file system, ZFS.[14] ZFS hasn't yet generated the same level of buzz as have DTrace and Containers. ZFS wasn't part of the early Solaris Express program, so users haven't experimented with it as they have with other new components. And file systems aren't the sort of component that users are likely to adopt instantly.

ZFS combines a new 128-bit local file system with integrated volume management. The 128-bit part serves up some strikingly high capacity limits, but that's probably less interesting to most users, especially in the near-term, than other elements. The integrated volume management, for example, redefines the role of the file system and, by so doing, simplifies disk management.

Historically a file system was associated with a single disk. As the number of disks connected to each system grew in number, volume managers were introduced to intermediate between physical disks and logical file systems. Because a volume manager presents itself as one large virtual disk, file systems could easily span multiple disk volumes without any changes to the file system code itself. That's the good news. The bad news is that administrating those multiple management products can get complicated. In addition, file systems still communicate with their underlying volume managers just as if they were communicating to a "dumb" disk drive—basically it can only tell the volume manager to read or write a block.

ZFS rethinks the entire "file system plus volume manager" architecture. With ZFS, there's no longer a volume manager at all. Rather, it incorporates the concept of a storage pool. This effectively works like a virtual memory manager (VMM) for storage; the pool delivers storage on request, just as a VMM delivers memory. In some ways, this is still similar to a file system and volume manager combination. However, by fusing the components, actions take fewer steps and are therefore faster and are less error-prone. Sun gives this example: to create a pool, to create three file systems, and then to grow the pool takes just five ZFS commands, as opposed to 28 steps with a traditional file system and volume manager. At present, ZFS is managed by command-line interface only, although Sun says that a GUI is a future possibility.

ZFS also has a 128-bit capacity, with far greater limits than the 64-bit file systems that are the standard in today's commercial OSs—such as Sun's own UFS in Solaris—and much, much greater than older 32-bit file systems. A 64-bit file system can support up to 16 exabytes (2^{64} bytes) of storage—an impressive number to be sure, but one that could start to become a constraint in the coming decades. In practice, real-world file systems have other constraints that keep them from reaching their theoretical maxima.[15] ZFS will doubtless have its own testing and qualification limits as well, but subject to that caveat, going to 128-bits for its new file system removes any architectural capacity constraints.

13. One can use DTrace on Solaris 10 to optimize an application and then run that same application on an earlier version of Solaris, or sometimes even on Linux and other Unix flavors. This allows existing Solaris users to take advantage of Solaris 10 before throwing the big changeover switch in production, and could help even shops who are not mainline Solaris users.

14. ZFS once ostensibly stood for "Zettabyte File System," but Sun says that the acronym now doesn't officially stand for anything. (A zettabyte is 2^{70} bytes, a number that doesn't correspond to anything in ZFS besides generic bigness.)

15. For example, Microsoft's "64-bit" NTFS could handle volumes up to 16 exabytes, in theory. Other hard-coded limits, however, currently keep an NTFS partition from growing larger than two terabytes. Even more than internal limits, the cost of storage devices imposes very practical barriers.

ZFS has other "foundational" aspects as well. For example, it has end-to-end error checking to ensure data and metadata integrity. Sun claims *19* nines reliability—a number almost as inconceivable as its capacity limits. ZFS is also "endian-neutral"—meaning that a disk can be moved from a system whose processor uses little-endian byte ordering (such as x86) to one whose processor uses big-endian byte ordering (like SPARC). Traditionally such a migration would have required backing up the disk, reformatting, and then restoring from backup—a disruptive and potentially data-destructive event. This unusual capability increases the data compatibility between Sun's SPARC line and its increasingly important Opteron line.

One thing ZFS is *not* is a shared file system that spans multiple servers. Given the ascendancy of clusters for high performance computing and other tasks,[16] shared file systems have been more a recent industry focus than local file systems have been.[17] Indeed, in its initial release, ZFS won't even support Solaris clusters, which will continue to use the existing UFS.[18] However, ZFS is designed to run particularly fast in conjunction with NFS v4, another approach to multiple systems sharing files across the network. Sun says that it is also considering enhancing ZFS to provide remote replication services for disaster recovery. For customers who require a shared file system for high-performance computing, Sun does offer a separate product, QFS.

ZFS eliminates many of the configuration and size limits of Sun's current UFS file system. While of long-term significance, these limits don't concern most mainstream customers today. It is ZFS' promised management benefits that speak more directly to the day-to-day concerns of today's IT staff. To be sure, achieving these benefits would require changes in administrative processes, approach, and even vendor[19]—which don't come quickly or casually. However, if the advantages are as close to what is suggested by Sun's characteristically breathless marketing, plenty of Solaris users will be lined up to at least put it through its paces.

Creating Trust

Like other Unix vendors, Sun has long had a separate "trusted" version of its OS. Sold primarily to secretive three-letter government agencies like the NSA, these special releases went through long and arduous inspection, review, and certification processes. Historically, they've also imposed so many mandatory security features and procedures (such as password aging and mandatory access control) that users routinely dislike them and mainstream commercial accounts find them more trouble than they were worth.

However, with servers more connected and with security needs greater than ever before, vendors like Sun are increasingly mining IP from their trusted OSs and transplanting or adapting it for their commercial products. With Solaris 10, Trusted Solaris will be largely merged into the mainstream product. While Trusted Solaris will still be a separate release, it will largely take the form of a "layer" on top of standard Solaris, likely with some certifications and configuration defaults specific to its government-segment customer base.

While Unix developers have had a general concern with, and paid attention to, security issues going back *decades*, Sun started to incorporate significant Trusted Solaris approaches with Solaris 8 and Solaris 9. The company began, for example, to embrace more sophisticated access models in place of Unix's traditional, rather binary approach to system privileges. Historically, someone was either an ordinary user with few, if any, administrative privileges, or an all-powerful *superuser* with access to pretty much everything on the system.

16. See Illuminata report "POWERing the Performance Factory" (November 2004).
17. See, for example, Illuminata reports "VERITASí Shared File Foundation" (September 2004) and "IBM SAN File System" (July 2004).
18. UFS remains part of Solaris. ZFS will be an additional option, as opposed to a replacement for UFS.
19. Many Sun shops today use VERITAS for their volume management and even file systems. Shops using VERITAS across multiple vendors' systems are likely to be especially slow to seriously consider ZFS.

Solaris 8's 1/01 update, and later the Solaris 9 main release, introduced role-based access control (RBAC) from Trusted Solaris. RBAC provides a means of restricting administrative access, allowing an individual access to only those commands appropriate to his responsibilities and expertise. Users log in through a preconfigured Pluggable Authentication Module (PAM) and can then assume authorized roles in role-aware shells. RBAC thereby offers a much more granular and secure alternative to the standard Unix and Linux approach of broadly granting full administrative access to root users. However, RBAC may still run a process as a superuser account in order to access all needed resources.

Solaris 10 now adds Process Rights Management (PRM) which can be used to restrict processes to having only the capabilities they need and no more. With UNIX, many processes typically run with root[20] privileges because they need some capability of the traditional superuser role. Unfortunately, this lets them access and modify most of a system's resources, whether I/O devices, files, areas of memory, etc. In other words, in order to get any superuser privilege, they get pretty much everything.[21] Malicious hacks often depend on this ability by subverting or taking over a process using a buffer overflow or other code vulnerability. Even in the absence of security breaches, root users or processes can cause unintended damage—such as deleting critical system files—through simple errors or coding mistakes.

Process Rights Management makes superuser privileges granular, ending the all-or-nothing proposition. For example, with PRM, there's a privilege that lets a process change a file's owner user ID. There's another that allows a process that is not the owner of a file or directory to modify that file's or directory's permission bits or ACL.[22] Still another allows a process to turn auditing on or off or modify the audit parameters. In all, PRM breaks the traditional superuser into several dozen individual privileges. Thus, individual processes can now be given the permissions that they need to work without giving them godlike powers over the entire system. A process that needs to change user files can now be granted that right—without letting the process change kernel parameters, turn off auditing, or do other things that it has no legitimate need to do (but which a hacker who has gained control of the process might well desire.)

Developers can specifically write applications that use or expect privileges, but it's not necessary. Applications that don't take advantage of PRM just default to the traditional way of doing things. If they run as root, all privileges are turned on and everything works the same as it would without PRM. Solaris 10 also provides a utility (`ppriv`) that helps customers and ISVs determine which privileges a process needs, so that an administrator can let existing applications take advantage of PRM without modifying them. Some traditional Solaris daemons and utilities have been converted to use the PRM model, but others continue to run as root. They will be converted over time.

The Open Future

Sun unabashedly promotes Solaris 10 over Linux, even though it also sells plenty of Linux on its Opteron servers. However, Sun also recognizes that Linux is here to stay, and that many customers prefer Linux to Solaris x86 for a variety of reasons, including application availability. To deal with the application issue, Sun has been actively building up ISV support for Solaris x86. Solaris 10 also debuts a new kernel service (codenamed "Project Janus") that lets Linux applications run unchanged on top of Solaris 10. Sun says there should be minimal overhead—about 5 percent compared to running natively on Linux. They can run within a container,

20. For the purposes of this discussion, a superuser and the "root" user are essentially synonymous.

21. Over the years, Unix has evolved approaches such as "effective user-id" as ways to temper its earlier all-or-nothing inclinations. But these are really workarounds that are subject to complexity, errors, and occasional exploitation, rather than a true Least Privilege framework like RBAC and PRM.

22. Access Control List, a mechanism for determining who is allowed to read, write, and execute a given file or directory.

but don't have to. Sun views this capability as a migration tool, a means to run Linux applications that don't have a Solaris version available, rather than a long-term alternative to porting apps specifically to Solaris.

Sun has also committed to open-sourcing Solaris in the near future.[23] The details so far are scanty, although Sun has been quite clear about a few things. For example, they've been quite clear that they plan to open-source all of Solaris—including new Solaris 10 features like DTrace and ZFS—and not hold back any of the "good stuff."[24] However, the open source announcement will be separate from the November 15 Solaris 10 launch, which Sun is gearing to its commercial accounts rather than the open source community in places like universities.

Conclusion

Sun's been aggressively making early copies of Solaris 10 available through its Software Express for Solaris program. Its goal is to compress the time it takes users to upgrade to a new OS version by effectively starting the evaluation clock ticking long before the release of the production product. It's effectively an open beta program preceding the production release that's slated for the end of January, 2005. Sun says that 71 percent of its current Early Access customers plan to deploy Solaris 10 in production within the next six months. If this holds true, Solaris 10 deployment could be rapid indeed for a new OS version.

Solaris 10 certainly has an outsized list of new features and capabilities—more so than any recent HP, IBM, or Microsoft OS release, for example. There's a reason for that. HP has been more focused on the HP-UX port to Itanium and bringing that up to parity with its PA-RISC version; HP-UX also must compete for HP mindshare with Linux and Windows. IBM continues to enhance AIX, but it is likewise putting enormous efforts into Linux[25] as well as layered Virtualization Engine components that are not part of the OS *per se*. Sun, on the other hand, is single-mindedly focused on Solaris. It's going up against the other Unix systems and against Linux, too. This release comes at what is, in many respects, a crossroads for Linux. Its commercial flavors are physical products from a few companies, such as Red Hat, with real (and not inexpensive) prices, rather than ephemeral and unbeatable open source ideals.[26]

Sun's certainly put enough effort into this version. Solaris 10 has cost an estimated 3,000 staff years and half a billion dollars in development costs. That's a huge commitment, especially given Sun's current business situation. The company is making this effort because it has returned to the view that Solaris is one of its strategic linchpins, a key source of differentiating IP and control on the volume x86 systems that it increasingly emphasizes alongside its traditional SPARC ones.[27] No longer does Sun situate Solaris as an important but largely hidden foundation.

For the first time in recent memory, Solaris promises highly desirable new functions that you can't readily get except from Sun. Containers are compelling to administrators and operators; DTrace is compelling to developers and production shops; and the runs-great-on-Opteron attribute enables Solaris virtues even on inexpensive volume gear. Sun is holding Solaris 10 up for all the world to see as a shining example of Sun's technological capabilities, what it can accomplish, and the advantages it offers.

---

23. See Illuminata report "Solaris Open Source Gets Serious" (August 2004).
24. Although Sun doesn't plan to hold anything of its own back, Solaris—like other OSs—contains technology and software licensed from third-parties. It seems likely that some such components, such as drivers, would be either held back or released only as binaries.
25. To say nothing of z/OS, i5/OS, and i5/OS.
26. See Illuminata report "Open Source Incivility" (April 2004).
27. There hasn't been much action around entry-level SPARC systems at all recently, although this will likely change when the first systems based on the radically multi-threaded "Niagara" chip become available, probably in early 2006. See Illuminata reports "Sun: Better Computing Through Threads" (July 2003) and "Breaking Up The Microprocessor Monolith" (July 2003).