



Announcer You're listening to the Sun Microsystems Podcast Network. Welcome to another edition of *Innovating@Sun*, with your host, Hal Stern. Today's topic, *Solaris 8 Migration Assistant*. And now, here's Hal Stern.

Hal:

Hello and welcome to another episode of *Innovating@Sun*. I'm your host, Hal Stern, Vice President of Global Systems Engineering, and here today to talk about *Solaris 8 Migration Assistant*, a fairly slick technology that leverages the best of virtualization containers and Solaris 10 features, to mix 8 and 10 together and get something bigger. So, to join me in talking about the innovation and what's going on under the hood are Joost Pronk, who is the Product Manager for Virtualization in Solaris, and Dan Price, Staff Engineer and the lead on the Containers project in the Solaris Kernel Group. So, welcome to the show, guys.

Joost:

Thank you, Hal, it's great to be here.

Dan:

Yeah, thanks a lot.

Hal:

And I guess let's start by giving a brief explanation of what the Solaris 8 Migration Assistant is.

Joost:

So, basically, the Solaris 8 Migration Assistant is a technology that allows you to run Solaris 8 environments on a machine that can run Solaris 10. It actually has different components, one of which is the bit that actually allows you to do the physical to virtual migration from the 8 environment, so that you can capture it on your 8 machine and bring it over; and the other is sort of the Runtime component, where you can run this 8 environment unmodified on your 10 machine.

Hal:

So, you used the phrase "environment," not application, so in order to clarify a little bit – we talk about the environment, we're talking about picking up the entire image and moving it over.

Dan:

That's right. We make an image of your existing system, using any number of existing tools, like the Flash archiving tools, or really, whatever tool you like. And then, we can move that image over to the Solaris 10 system, and when you install the Container on the system, you pass the image as an argument to the installer, and it basically looks at it, unpacks it, figures out what to do with it and then lays that down on the system. And so, this gives you the ability to move your applications whole-hog, with all the settings and all the configuration files, right from one system to another.

Hal:

I was going to say, I think that we talk a lot about application migration, but that misses the larger whole of what really goes into deploying an application – libraries, files, configuration information – there's a ton of context information that's actually part of the environment and the Migration Assistant allows you to capture all of that, right?

Dan:

Yes. And when customers are doing migrations, all those things you mentioned they see as potential points of failure, and places where their human operators can make mistakes. So, we decided to try to skirt that problem by just saying, Okay, we're just going to make an image of everything and then set it down inside of the container.

Joost:

And I guess also in new environments, maybe some of those settings are different. There might be arguments are different, or if you have a scripted capture stuff, stuff might have moved around. So, all these things are places

where there's possibility for things to go wrong. In this particular case, all of that stuff just comes over with you and will run in the same way on the new system.

Hal:

Yeah, and I think there's nothing worse from the availability perspective of running an application that you've moved from one environment to another, only to find out that your system monitoring or your healthcheck tools aren't working now and so, the operations staff think it's failed when in reality, it's working just fine. So, this actually gives you, I would say, probably a slightly risk-reduced way of doing a migration. I guess the fundamental question is: Why do people want to do this? I mean, to me, it just seems like go and recompile, right? I mean, why would you want to migrate an entire OS operating environment from Solaris 8 to Solaris 10?

Dan:

Well, we had the same question when we started out thinking about this work, which was at the beginning of this calendar year. And what we heard from customers over and over again was that they have an awful lot of systems, in many cases, and they're really anxious to get to Solaris 10. They're really anxious to get to new hardware platforms like the Niagara-based platforms, because those platforms offer them a much greater economy of scale, reduced rack space, reduced power and cooling requirements, and even a reduced part count on the machines, which makes them more reliable. So, as a result, we found that customers were saying, If I have a thousand systems, it's going to take me a really long time to get them all from Solaris 8 to Solaris 10; and at least in some of the systems, I've forgotten, you know, how to install the applications on that system. So now, I'm going to have to go relearn how to install the system – I have to go figure out what the system is for, I have to hunt down the developers, or maybe they've left my company. So, we saw this as a way to provide a path to consolidation without so much cost to the customer.

Joost:

And I guess there's also another way, also looking at the same thing is, when they install the application in the beginning, say five years ago, they locked themselves into Moore's Law, or that point in time in Moore's Law five years ago. Now, stuff has moved on and they want to sort of put the application onto a new technology that has sort of emerged in the meantime, giving them all of these qualities of lower wattage per cycle and all these other types of things, and this gives them an opportunity to pick up the whole environment and just put it onto this later point in time of the curve.

Hal:

Again, let's be a little bit more explicit there, if you want to go run one of our newer servers or the M series servers or the Niagara Cool Thread servers, they're running on Solaris 10...

Joost:

Yep.

Hal:

...so you can't let the environment be an impediment to getting there.

Joost:

Mhmm. Yeah. And we've done quite a lot of innovation on Solaris 9 and on Solaris 10, and things have changed in the meantime. And even though the applications themselves will run there because of our binary compatibility guarantee, it's like you said earlier, the environmental around it that actually quite often are the things that have people scratch their heads, How do I do it, again?

Hal:

I think, Dan, you made this point briefly just before – once you have an application up and running and tuned and installed, your environment is the sum total of all the work you've done up to that point, not just a recompile of the source code. And unless you're very, very good about that mastering and provisioning process, it's hard to migrate. And so, you have the environmental capture here, I think takes some of the cost and risk out of that.

Dan:

Another point that surprised us when we were talking to customers was that a lot of them do their own software development, as you alluded to when you said, you know, they might recompile. And if they're supplying software to other parts of their company or to other companies, they may have to ensure that their software runs on Solaris 8, Solaris 9 and Solaris 10. And one way to do that is to actually create Solaris 8 Containers and use those as a development environment to do your compilation to Solaris 8. And for a lot of customers, this may allow them to more efficiently use compute resources in their developer farm of systems.

Hal:

And I think it's that deployment side of it that I'd like to dig into a little bit more. You talked about the physical to virtual tool, which essentially taps an existing Solaris 8 environment and turns it into an image that can be installed. It's actually being installed into a Solaris Container. And I know we've had a project called Brand Z – we went through a few name changes to avoid conflicts with '70s art rock bands, among other things. Why don't you tell us a little bit about what actually happens once you do the install of this virtual image into a container. What's happening inside that container that makes the dynamic environment here, from a Solaris 8 perspective, run on top of or inside of, as a guest, of Solaris 10.

Dan:

Okay, so hold onto your hats. I guess the easiest way to think about Brand Z is as I've started to say to people: It gives us the ability to create a pack of lies that we can tell applications about what kernel they're running on. Ultimately, applications in Unix environments interact with the kernel in a limited set of ways. One way is system calls, another way is signals. And there's a few other interactions that take place, but we use those points as places where we can put our interposition code. And we do that in two ways: One is, in the kernel, we have a special trap handler that can run that knows that a particular process is branded. And when that happens, we can do something called a trampoline, where the process makes a system call, we recognize, Ah, this is a Solaris 8 process, not a native process. Okay, we're actually going to then return from the system call into a special library of code that we've mapped into that user application. We return there, we can then run inside the application – we've sort of hijacked the application with a shared library. We run inside there, we do our interpositioning, we can make an arbitrary number of system calls back into the kernel and then when we're done, we return control back to the application at the point that it made the system call. So, we tricked the application into sort of lying to itself, if you will, because we map all that code into the application itself.

Hal:

I guess there are two very old operating systems techniques– you used the phrase “trampoline,” which if you've dug through kernel code in days of old, you remember the way that signals get handled, which is you basically, you know, pause the application, go do something else in the whole library of code, and then come back to where you were. So now, you're just able to do that across a broader range of operating system interactions. The other part is the phrase “interposition” which, I think, has been part of the Solaris vernacular since we started doing dynamically-linked libraries – the ability for you to go load different versions of a library into an application at runtime – in some cases, to affect different behavior. In this case, you go affect a different environmental interface.

Dan:

Indeed. Yeah.

Hal:

So, it's not to dismiss what you're doing – this is not new stuff in Solaris, it's simply a really radical application of some existing operating system concepts.

Dan:

Indeed. We use a slightly different mechanism for the interpositioning than the traditional shared library interpositioning, where you would LD preload something in front of the application – although it's very similar. In essence, when the process execs, we map a separate copy of the linker and a separate copy of the brand library into the address space of the application, and then, when we start the application at the end of the exec system call, we pass control directly to our linker, which then links and transfers control to our brand library. Our brand library gets to run at the beginning of process startup, and it ultimately transfers control back to the processes linker, which then runs. So, it is actually very similar to that mechanism; it's just that we've used some hooks in the kernel to make sure that there's no way to sort of defeat that interpositioning – you can't accidentally set a linker environment variable and cause the whole works to come apart.

Hal:

And once again, I think that there's certainly been some discussion lately that virtualization and security don't always appear in the same sentence, and I know there's a lot of work that has been done in the dynamic linking world, making sure that libraries don't get interposed, either accidentally or maliciously, to introduce undesired consequences.

Dan:

Right. And I think it's worth noting, too, that because we moved all of our interposition code into user land for the

application, it doesn't pose a security risk. The application, as far as the kernel is concerned, is still just a Solaris regular, old application; and so, all the security work that we've done on Containers still applies.

Joost:

Yeah, I think it's also good to kind of point out that because the whole thing is actually running in a container, the privileges that we've actually reduced when you run in what we call a non-global zone, all of that applies to this. So, all of the things that you would have naturally, like not being able to talk to kernel memory, or one of the default things is, for example, you can't change your IP address. Those type of security lock-down mechanisms that we have on the container apply here, too, naturally, and you just get them out of the box, even though it wasn't necessarily the case in the Solaris environment.

Hal:

And how far abreast of the actual OS interfaces does the Container infrastructure go? I mean, do you actually have to worry about things like different file systems name space in slash user and other places, are there other things that get translated along the way?

Dan:

One of the things that's very elegant about Brand Z is that we don't have to do any file system name space translation. And this is actually different than some of the kind of previous generations of technology that have appeared in the space. The nice part about the container is that it takes care of saying what the root directory is for applications. And so, in essence, in the kernel, we know what the root directory is for each and every process on the system – that's actually something you can set, that's how the old Chroot mechanism works, it actually alters the root directory. And so, we use a similar mechanism for containers, and that applies here, as well. So, we actually don't do a lot of lying to applications, we don't translate path names from one to another – which is actually really great, 'cause that's kind of a hard problem.

Hal:

And again, not only is a hard problem, but it's one that presumes you know all the answers in advance when you go and do the software release, versus, Gee, someone who's in deployment or a system administrator may have gone and made a change. So, it's as late a binding to the file system emphasis as you can get, by capturing the entire environment. So, we talked about Brand Z, I guess, here in passing, as something that was built on top of Solaris Container technology, and I would say we first started hearing about Brand Z in conjunction with being able to run Linux applications inside of Solaris. So, I don't know if you want to take a bit of a detour and give a bit of the backstory on Brand Z here, in terms of where you see Brand Z being used more generally as a guest operating system virtualization facility.

Dan:

Sure. That'd be great. I really like Brand Z. I confess that I wasn't on the team that built it, but I've really enjoyed learning about it, and that was sort of the first thing that I did as part of this project, was go figure out, Okay, what do we have here with Brand Z? And what I discovered is that we really had a set of hooks that allow us to alter the behavior of containers. And it's pretty remarkably easy to make a very simple brand that only really – for example, let's say you wanted to have a brand where you used a different packaging system inside of the brand. That's really easy to do, and you wouldn't have to write a single line of kernel code to do that. When the brand installs, there's an install hook that's described by the brand's description file, and that says, Well, what program do I run to install a zone with this brand? And so, you simply would then supply a shell script or a program that would do that installation. And so, it's kind of a way of making containers more generic by getting rid of hard coding sort of a whole bunch of default behaviors, and letting you override those.

Joost:

And I think the other interesting thing is that initially, Brand Z was focused on investigating if we could actually run a more fully-fledged Linux environment on top of Solaris on the x86 platform than what we had with LX1 [?], where you sort of – yeah, you can run the Linux app but all kinds of environmentals are missing, and what if you want to take the whole environment over and install the whole thing whole-hog, but just sort of neglect the kernel pieces where needed. And that was actually the original sort of goal of this thing, but what we figured out is that it's so universally usable, we can also use it on, in this case, the SPARC side and do Solaris 8 SPARC, because there's a lot of Solaris 8 SPARC out there and not so much x86 Solaris 8. So...

Dan:

[overlap] We actually had to do a bunch of work to it to make it work properly on SPARC as completely as we needed to for the Solaris 8 case, but it's always been something that's been a generic to both platforms; it just turned out that we had to lay some extra track to make it work, to do all the tricks we needed to emulate Solaris

8. So, that's good, because now, we really have the facility truly generic to both platforms.

Hal:

And I think what we end up with is a composition of virtualization techniques. If you go all the way down to the hardware level, we have L-DOMs on the SPARC side; we have Zen and VMware – on the x64 side; there's Solaris Containers as a common unification point at the Solaris operating environment; and then, our ability to go use, on the SPARC side, the Solaris 8 Migration Assistant on the x64 side of Solaris, to go use Brand Z to bring in Linux environments. I mean, you have the ability now to think about where is the layer at which you want to start virtualizing your resources? Is it at the OS environment, or is it the power of virtualization hardware layer, to go effectively allocate resources. And I'd say, more importantly, to go control the management of those resources, to go establish a single – or as few points of control as possible.

Dan:

Yeah, you can even compose some of those technologies together. So, people often ask us, Well, could I run a Solaris 8 Container inside of an L-DOM? And the answer is, Yes, of course.

Hal:

And I think that's certainly where people would like to go with this, is to be able to look at not just migrating the environments one for one, but finding an economy of scale in terms of perhaps compressing multiple containers into the same global zone, or into the same virtual machines for L-DOMs, or physical machine; or in other cases, keeping that mapping of one app equals one virtual environment, because that's how they're used to managing and provisioning them.

Joost:

Yeah, and they sort of get to choose where they do the sharing, how high up in the stack they start to kind of say, Well, this is all shared and above that, we're splitting stuff out. So, for example, on the container side, your volume management and your network management is shared and you can set that up once and every container can live on your ZFS pool, whatever you set up in the global environment; whereas, if you go more to the virtual machine side, you get to choose. You sort of get more choice – you have to manage more because you have choice, but you know, if you need to have that choice, you get to choose what you want.

Hal:

In many cases, I think that – those choices will be governed by what the granularity of provisioning is, in terms of network bandwidth or application isolation and security or how provisioning is done. How is it that the system administrators look at stamping out more instances of the application, or multiple applications, and they'll use, essentially, their Runbook practices to govern which of the multiple facilities they're going to go use and in what sequence and what composition. Which, I think, raises an interesting question. So, where wouldn't you use Solaris 8 Migration? It sounds like a great technology, but there has to be, probably, some fair warning here. Where would you not use this?

Joost:

So, let me take a first stab and Dan can sort of add to this. I guess the key thing is, we didn't design it to have this meet all needs, because one of the things we actually found out when we started to talk to customers about this is that there's actually a certain subset of their applications that have already migrated that are so important to them that they actually want to customize performance too, and all these other types of things. But, I think one of the biggest Aha! Erlebnisse for me was, there's actually a whole bunch of applications that they don't want to spend the time on touching and that still need to migrate over to this new hardware. And so, we've already focused on those and having a look at what those are; and quite often, those are fairly simple applications, where most of the stuff is user land, there's hardly any pieces that lock into the kernel or do things like that. You know, I'm just joining a printer server or a simple database or mail server, and it's doing a singular task that someone had envisioned, but is not doing something very complicated and sophisticated to the system. For that, it works very well. The moment you do get the sophistication and you have certain kernel components, or maybe pieces that you need to twiddle more effectively, this wouldn't fit very well. And I don't know if Dan wants to add some pieces.

Dan:

Well, I think that there are definitely some limitations and we can definitely talk through those. The first thing that sprang to my mind, Hal, when you asked the question was that what I would say to folks first is, try Solaris 10 Containers first. See if your application works perfectly inside of a Solaris 10 Container. If you're frequently deploying that application and you know all about how to install it and you've got a script that drops it onto a system, just run that script on your Solaris 10 Container and see how well it works. It will probably work

flawlessly. So, I guess my bias would be that we want to get people into Solaris 10 Containers and really reap all the benefits there, and there are a lot of performance benefits that exist in Solaris 10 and while Solaris 8 Containers can take advantage of some of them, not all of them are available. So, we're not trying to encourage people to run Solaris 8 until the end of time, and we really see this as a transitional step between Solaris 8 on raw hardware, into Solaris 8 on a Container, to Solaris 10 in a Container. So, I guess that's what I would say about where it doesn't really fit; but certainly, if you're facing a really huge migration task, I think this can be really helpful and also, in cases where Solaris 8 is mandated for some reason, I think this can be really helpful.

Hal:

Several of the things, I think, that make Solaris 10 attractive to both developers and deployers are features like ZFS and the file system storage pulling, as well as Dtrace from a performance and dynamic profiling capability. If we use Solaris 8 Migration Assistant, does the Solaris 8 application get to take advantage of those Solaris 10 features?

Dan:

It does, actually. It turns out that we can use ZFS as the underlying file system for your Solaris 8 Containers. Also, from the global environment, the global zone, you can run Dtrace and observe all of those Solaris 8 applications that you have. And there's actually a raft of other Solaris 10 features that you get to take advantage of, as well; for example, we've got the FMA default management architecture infrastructure that is watching for potential hardware issues on your system – that stuff is guarding your Solaris 8 instances, as well. Another example would be the resource management work that we put into the operating system. That wasn't really there in a really coherent way in Solaris 8; but now, if you want to limit a container to three CPUs, that's really, really easy to do. We actually did a lot of work in Solaris 10 8/07 to make things like that really very simple. And so, you kind of get this funny mix of both worlds, and you get to take advantage of some of the Solaris 10 advanced feature set.

Hal:

Great! So, you guys know that I'm on the sales side of the world here, so I have to ask: Where do people go to find out more information?

Joost:

So, one of the best places to start is Sun.com/Solaris/Containers. There's a Learning Center there and there's a link there. There's a link to being able to download it and there's all kinds of materials there. I think the other thing that would be very good is to go to your local Sun sales rep or someone who sells Sun and go talk to them and get yourself a demo. But, you can actually download the bits off the web, you can try it out yourself. There's a 90-day free evaluation license with this, so go knock yourself out. And you'll be, hopefully, interestingly surprised in how it works.

Dan:

We were really surprised and happy when we ran the numbers and discovered that, really, a lot of people have downloaded it already from all over the world – as far away as, you know, Ecuador, or as [laughs] – so, it's pretty cool.

Hal:

Great. Well, exciting stuff. And I would say a little bit of old school OS theory and a little bit of looking forward at virtualization coming together to create a really robust migration plan. One of the things we talk about frequently on this show is how we protect investment and how we provide migration and I think this is a tremendous example of using some really innovative technology to ease our customers' migration to where we're going, without forcing OS disruptive points or OS flag days along the way. So, Joost and Dan, I want to say thank you for joining us. And you've been listening to another episode of Innovating@Sun. I'm your host, Hal Stern.

Announcer You've been listening to Innovating@Sun. Join us next time for the latest in innovation from Sun Microsystems – only on the Sun Microsystems Podcast Network.