From: Dan Neault
Sent: Tuesday, April 08, 1997 11:23 AM
To: Cory Van Arsdale (LCA)
Subject: RE: BeOs

Privileged

-----Original Message-----
From: Cory Van Arsdale (LCA)
Sent: Tuesday, April 08, 1997 10.33 AM
To: Dan Neault
Subject: RE BeOs

Privileged

-----Original Message-----
From: Dan Neault
Sent: Monday, April 07, 1997 8:29 PM
To: Cory Van Arsdale (LCA)
Subject: FW BeOs

-----Original Message-----
From: Mark Lucovsky
Sent: Monday, December 16, 1996 4 35 PM
To: Dan Neault, Frank Artale, Eugene Ho; Lou Perazzoli, Sudeep Bharati
Cc: Mark Zbikowski
Subject: BeOs

Markz and I are still not sure what exactly you are looking for. Given the time pressure you are under, we decided to supply you with a random set of observations we have.

Everything we know about BeOs comes directly from their website. I don't think they lie on their website Their info seems pretty straight.

My key observation is that they have extremely immature technology Anyone betting on this as a strategic way of getting somewhere quick is in for a big let down.

- Their OS is design for SMP. This is great, but since it was built on the BeBox SMP system, I would fear that the UP performance is poor and that they really need SMP in order to run. There is a comment in a benchmark FAQ that eludes to this. It says that perf degradation on UP happens quickly and that SMP is better. I would be afraid that BeOs needs extra CPU to support their rather heavy weight architecture.
- The claim to be similar to Unix on the insides but claim their kernel is preemptible and supports nested interrupts but that they don't currently use nested interrupts. I find this hard to beleive. I think they have such limited device/configuration support that they really don't know what they support.
- They are building an OS but claim they will depend on others for key support like support for Iomega Jaz. We know how hard getting broad support is and I have a feeling they have a very weak architecture that will put a huge burden on some of these ISVs. Seems very risky, but from reading the docs, I get the feeling that they are stretched very thin and just don't have time or resources to do a proper architecture.
- No UNICODE, no DBCS. This is incredibly limiting. In reviewing the API set, adding this later does not look easy. Again, this is a sign of poor architecture planning.
- They have a very poor architectural structuring with nothing that resembles a HAL. This gets into their I/O architecture. When hardware advances occur, they will be slow to capitolize on the advances. I don't think they have done a good job abstracting the machine enough to support new innovations. Other areas that are affected by this are things like booting. Since they have a poor abstraction, they need to staticly link the boot drivers into the kernel. This of course means that booting on a new device is out of the question until the kernel revs. Since they were so tightly coupled to the BeBox this was not a problem since they defined the hardware and boot firmware. The switch to Mac was a demo where they boot BeOs by running a MacOs app that takes over the machine as BeOs. The final step to boot on Mac is to link in the small set of Mac boot devices.

- There is no file shareing. Their whole filesystem seems pretty weak. One comment makes it sound like after 10,000 files things start to fall apart. File sizes are limited to 2Gb/file with 63 char file names. Seems very limiting, but more importanly again highlights the architectural immaturities.
- Their MP support seems incredibly weak. I can not imagine seeing any form of decent scaling. I hope they are not pushing this is a platform for demanding workstation uses.
- I saw references to polling loop programming with calls to Sleep() (actually snooze()) to yield the CPU. This seems odd since they are really message driven. Again, I think this is just a weakness in their overall architecture. They are building a toy OS. It would never stand up under highly stressfull conditions and is probably easily resource starved.
- No security. None. No object protection. Everything seems to have a global name including semaphores, threads, vm regions... I can see very easily how a programming error in one application could tank another application. They have address space seperation, but making all objects globally visible is incredibly dangerous. Of course it is easy to do and once again highlights the OS immaturity.
- Their threading model is incredibly weak. It really represents a research study in building a simple threading system. It could never handle the demands of big systems or highly controlled threaded systems. There is no processor affinity, stack size control... I could not find evidence of exception handling. Their attitude really seems to encourage threaded systems easily without worrying how to solve some of the hard problems or provide highly robust primitives.
- The synchronization APIs are a joke. They have semaphores. They recently added a timeout value to the wait. This again points to the immaturity of the design and their lack of understanding One of our biggest strengths is the technical bredth of our API set, the fact that it has been reviewed by many top designers across all slices of the industry Their API set doesn't even look as complete as our initial Win32 draft that we previewed on that snowy day in december 1990.
- Their API is very mach-like with ports and message passing. Again, they recently added asynch sends. I get the feeling that they band aid the API without thinking through exactly what they are trying to do.

I guess I could go on and on. The bottom line is that I beleive they can make a very compelling snazzy looking demo. There is no question in my mind that their technology works and will demo well I think their OS is weak I think their API set is very immature and does not offer the richness needed to solve problems faced by broad classes of both desktop and back office applications. I think they have a poor internal architecture and will not be able to take advantage of hardware advances as they occur. Asynch I/O or other very high end features that are crucial for Web servers or other demanding CS apps appear to be missing. Bolting these in as an addition will likely invalidate ALL existing BeOs drivers. Their driver model seems very simplistic and flat. I don't see how they reliably do things like layer in FT.

The os is very immature. It is likely built by a small team and has not had broad serious review. It might be fine as a multithreaded lightweight windowing system. Sort of a notch between Wfw and Win95.

Other things that seem to be missing are interop architectures like OLE or OpenDoc. Maybe they spin this as a positive, but these are notoriously complex things to build and require alot of architectural disipline. I don't know where they get this or how they build it.

Good luck tommorrow. Email me if you need more info. I am sure markz will add some comments as well.

-markl