connect
OpenVMS
Boot Camp 2011

September 18 – 22
Sheraton Hotel – Needham, MA

Developing Avanti and FreeAXP Virtual Alpha

Session I313

Camiel Vanderhoeven

Disclaimer

- The following represents Migration Specialties International's current view of its product development cycle and future directions.
- It is intended for information purposes only, and should not be interpreted as a commitment on the part of Migration Specialties International.
- This document is subject to change without notice.
- Migration Specialties International makes no warranties, express or implied, in this document.

Session Overview

- Starting point: ES40 Emulator
- FreeAXP
- Avanti
- Roadmap
- Under the hood

Starting point: ES40 Emulator

All glory comes from daring to begin

Eugene F. Ware

ES40 emulator: what is it?

- Open source, developed by Camiel Vanderhoeven
- Written entirely in C++
- Multi-platform (Windows, VMS, Linux, OS/X builds reported)
- Boots VMS
- Support for text-mode VGA console

- www.es40.org

ES40 emulator: drawbacks

- Slow
- Buggy
- Terribly complex, slow VGA console emulation without much advantage over a serial console
- Complex system picked for emulation
- Dependent on hobbyist efforts, no model for generating revenue

FreeAXP

FreeAXP: changes from ES40

- Simpler Alpha system picked for emulation (AlphaServer 400)
- No graphics emulation
- ShadowCode technology for a much faster (~2x) cpu
- More efficient use of host cpu cores by smarter scheduling results in faster, more reliable I/O (network, disk, serial port)
- Free for any use, but not open-source
- Windows only

Avanti

Avanti: Advantages over FreeAXP

- Built from same code-base
- Native instruction translation results in higher cpu performance (~3x FreeAXP currently, limit not reached)
- More emulated RAM (3GB) and devices (NICs, disks)
- Support for physical hard drives and serial ports
- Starting in version 2.1, multiple Alpha systems can be emulated on a single host system

Roadmaps

There is more to life

than simply increasing its speed

Mahatma Gandhi

Virtual Alpha Roadmap

- 2.0.1: current
- 2.1: multiple instances of the emulator on a single system
  - Currently in test
- 3.0: EV6 cpu support
  - Custom firmware
  - Wider range of supported systems
  - Faster CPU
  - More RAM supported
- 3.1: SMP
  - Faster processing with multiple emulated CPU's

Planned Virtual Alpha Patches

- 2.0.1: current
  - Fixes NIC issue with Tru64
  - Issue: Tru64 kernel panic related to NIC traffic observed on a small number of Tru64 installations
- 2.0.2: Faster SCSI controller
  - Current SCSI implementation deliberately slowed down to avoid mount verification issue on OpenVMS
  - Should show a 2-3 x improvement in disk I/O throughput
  - Currenty under development

Virtual VAX Roadmap

- 1.0: Beta version
  - Currently under development
- 2.0: Mayfair
  - Emulates a MicroVAX 3900
- 3.0: Cheetah
  - Emulates a VAX 4000 model 100
- 4.0: SMP
  - Faster processing with multiple emulated CPU's

Under the hood

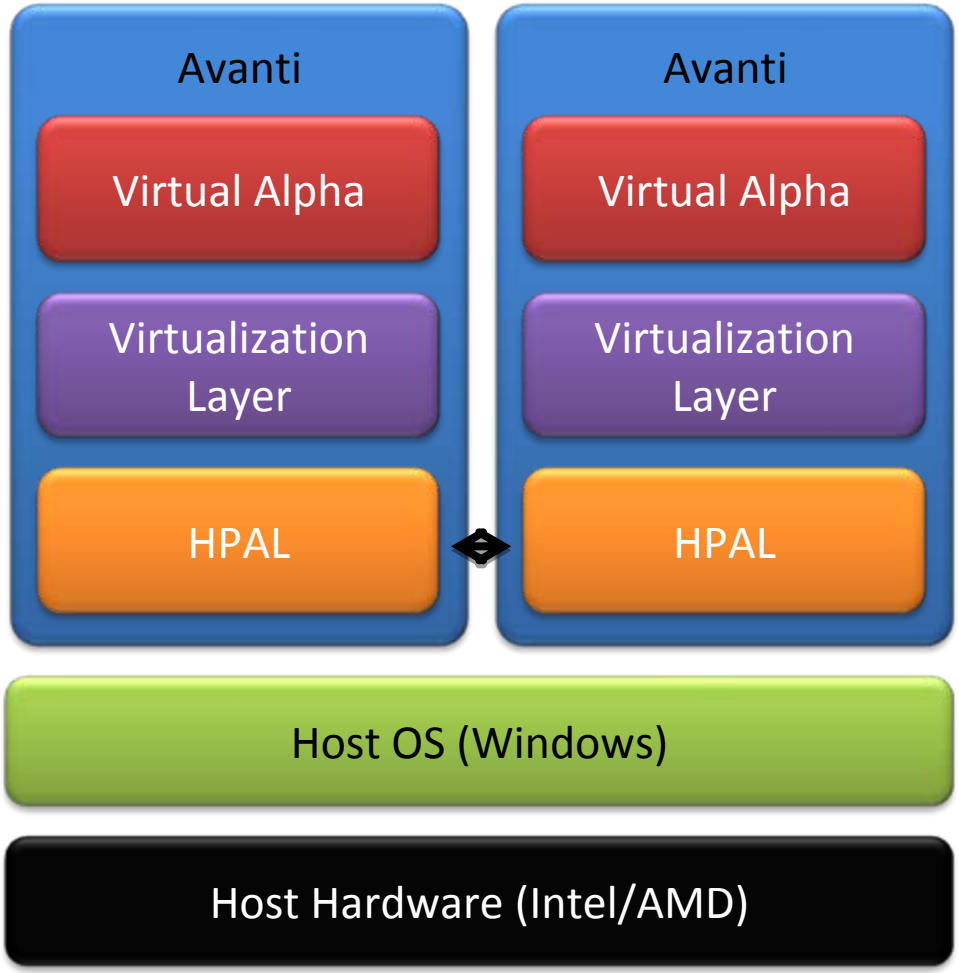
Beauty in things exists in the mind which contemplates them

David Hume

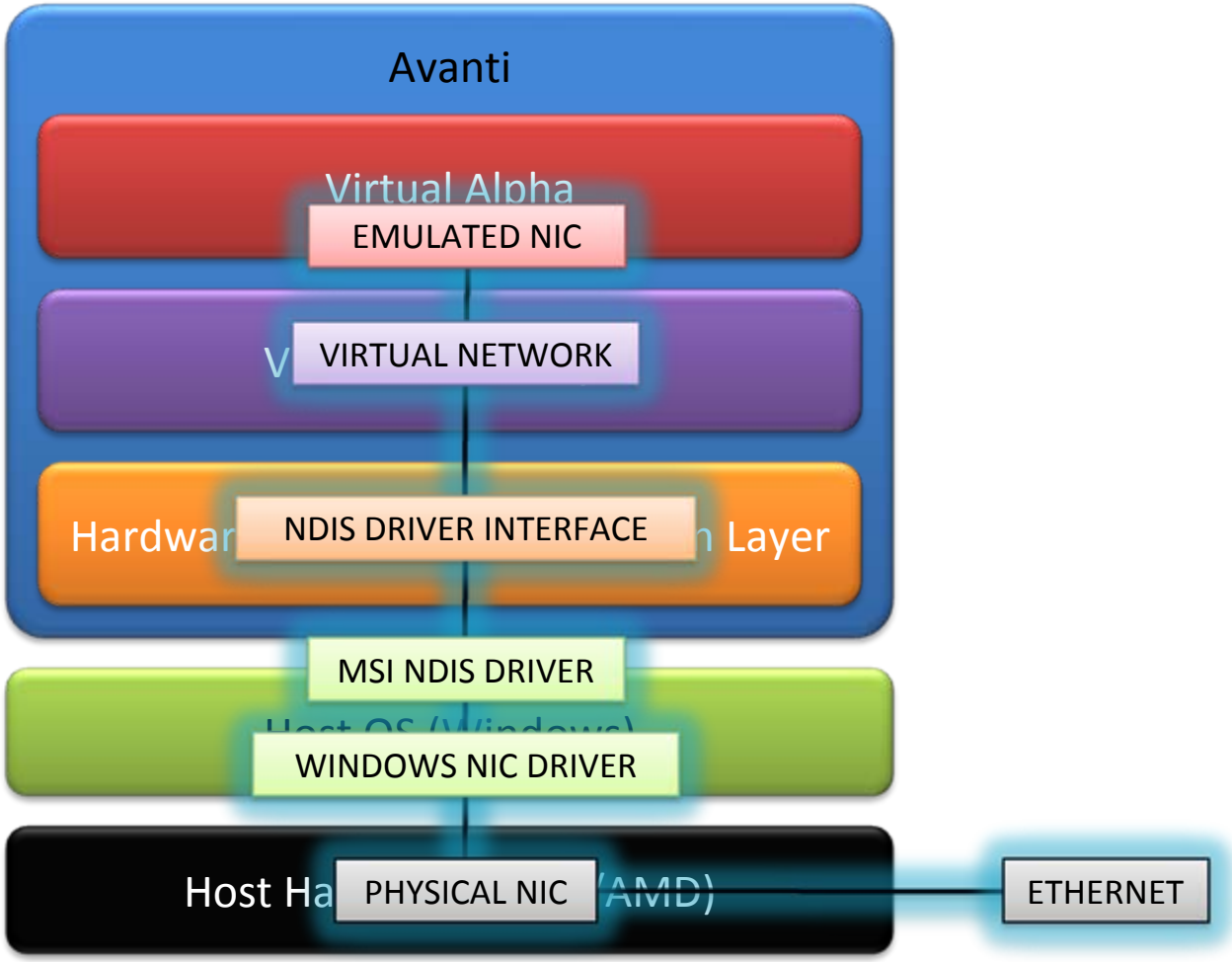
Overview: Under the Hood

- Overall emulator architecture
- Network architecture
- Concurrency within the emulator
- Toolchain

Overall emulator architecture



Overall emulator architecture: network



Managing concurrency

Time is what keeps everything from happening at once

Ray Cummings

Concurrency

In a real AlphaServer system, multiple components are active at the same time:

- CPU
- SCSI Controller
- SCSI Disks
- Network controller

These components perform their functions independently, and communicate with each other

Concurrency

Multiple ways to emulate this:

- Single thread does all the work
- Multiple threads for different components
- Actor model (message passing)
- Mixed thread/actor model

Single thread (e.g. SIMH)

All work is done in a single thread.

Advantages:

- Straightforward, easy to design and debug
- No locking necessary

Drawbacks:

- Slow (when the emulated cpu communicates with another emulated device, its function is performed before the cpu can execute the next instruction).

Multithreading (e.g. ES40)

Each device has its own thread(s). Data is shared between threads, this requires lots of careful locking.

Advantages:

- Much faster, cpu can immediately execute the next instruction

Drawbacks:

- Locking makes design and debugging difficult.
- Locking reduces performance.

Actor model

There are as many threads as there are cpu cores. Devices are implemented as actors that send messages. No shared data. (Erlang, Scala)

Advantages:

- Faster than single thread, cpu can immediately execute the next instruction.
- No locking needed.
- Very efficient scheduling

Drawbacks:

- Message passing can be expensive, especially when dealing with large amounts of data.

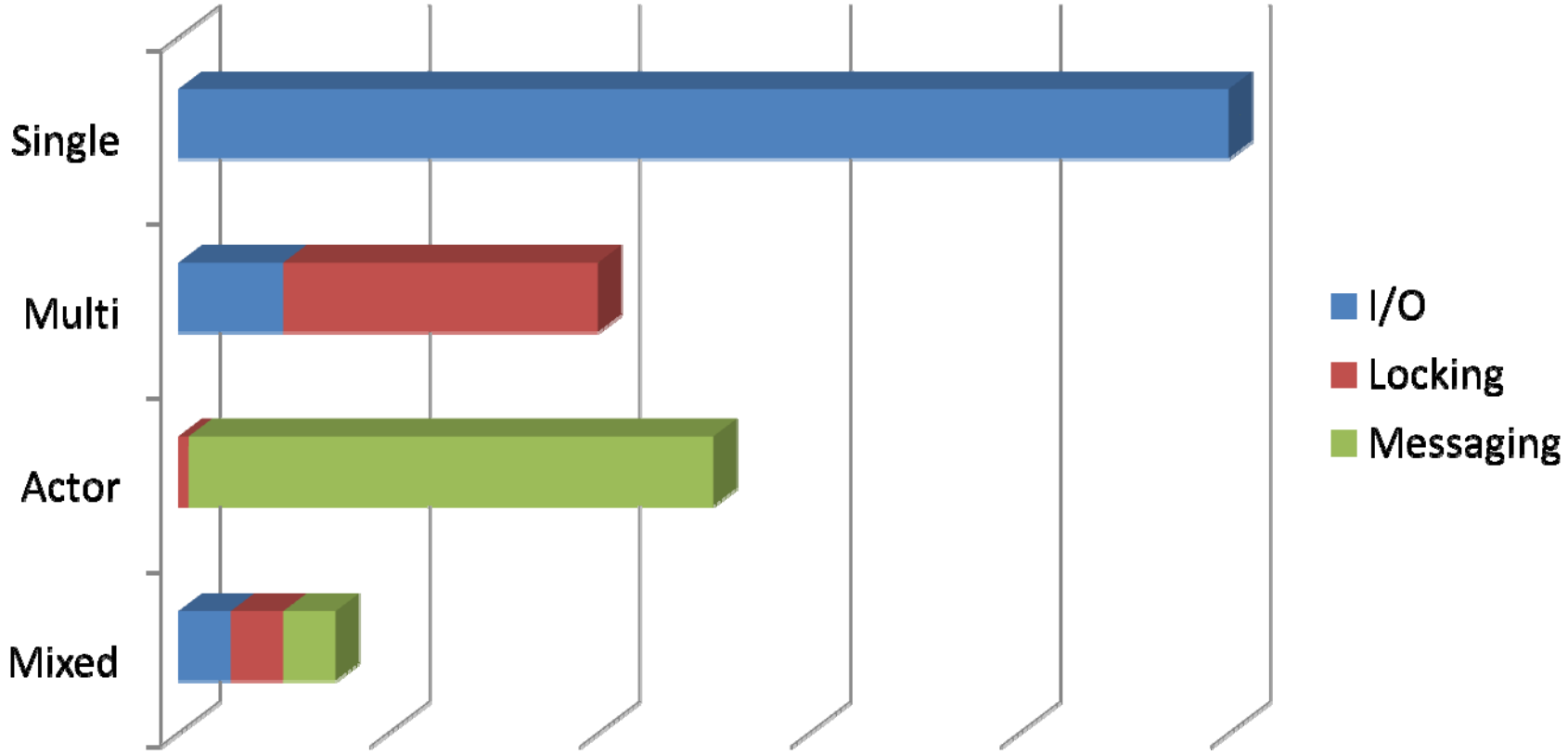
FreeAXP/Avanti mixed model

The emulated cpu has its own host cpu and thread. The other host cpu's are used for actor model threads.

Advantages:

- Fastest implementation (low locking and low messaging overhead)
- Very efficient scheduling

Overhead in CPU thread from I/O, Locking and Messaging



Toolchain

An architect's most useful tools
are an eraser at the drafting board,
and a wrecking bar at the site

Frank Lloyd Wright

Issue tracking: Bugzilla

Code repo: Mercurial

Issue tracking client: Deskzilla

Windows repo client: TortoiseHg

IDE: Visual Studio

Tuning: Intel VTune

Compiler: Intel C++

Installer builder: Tarma

Build automation: batch scripts

Automated testing: Python scripts