Linux Migration Plan

Table of Contents

Quick Outline

1. Create toolchain
2. Create build infrastructure and makefiles
3. Develop porting framework for developers
4. Linux bring up on Cougar
    a) Assemble filesystem
    b) Assemble kernel bits
    c) boot kernel on Bobcat TXRX processor
    d) boot kernel on Cougar hardware
5. Port EverON to Cougar
    a) Startup scripts
    b) root filesystem packaging
    c) upgrade utility
    d) management driver
    e) chassis driver
    f) GPP <--> TXRX TCP port forwarding virtual networking thing
    g) Port applications code
6. Linux bring up on Bobcat
    a) kernel config
    b) extra drivers, if any
    c) boot kernel on PMC processor
    d) GPP <--> TXRX TCP port forwarding virtual networking thing
    e) management driver
    f) chassis driver

Porting framework kit for developers

In order to keep the software development aspect of the OS migration manageable, we need to modify our current software in place, rather than creating a Linux-specific tree or branch and a OpenBSD specific tree.

To accomplish this with the most efficiency without sacrificing code maintainability, the plan is to port 2-3 of our applications and in the process create a basic framework kit that all the developers can use to make the applications portable. This kit would include common wrappers for things like system calls and C library functions that have differing semantics, as well as commonly used APIs that might be changing due to differences in hardware or OS. In addition, this kit should also include Makefile frameworks for making our build system portable and simple to use. Modifications to our Makefiles will be a central part of a portable software base and build system.

A wiki write up and an informal class on how to use the framework items and how to create new ones will give us a jump start into making all our applications portable in time for alpha product schedules (mid 2007).

The idea is to create this early in the migration process so that porting of GPP applications can run in parallel with bring up and other efforts.

Linux bring up effort

Step 1: Create reproducible new toolchain

(a) gcc

(b) glibc

(c) binutils

(d) gdb/kgdb

Step 2: Assemble file system

(a) build file system

(b) file system build makefiles

Step 3: Assemble kernel bits

(a) get latest bits from linux-mips.org

(b) configure device drivers

(c) console driver

Step 4: Modify kernel for our hardware

(a) memory map

(b) tftp boot of debug kernels

(c) tftp boot of kernel + NFS root

(d) CF boot of kernel + NFS root

(e) CF boot of kernel + ramfs/unionfs/ext3

(f) debug capability

(g) control NFX processors

Step 5: Early bring up on Bobcat TXRX processor

The plan currently is to start bring up efforts by making a limited effort to bring up Linux on the TXRX processor on a Bobcat. This processor is quite similar to the ones that will be used on the Cougar platform, and will be a good early testbed. Cougar hardware for OS bring up will not be available until February 2007 at the earliest, so there is some time between now and then.

Step 6: Bring up on early Cougar hardware

(a) Hardware and software bring up and debug

(b) CF driver

(c) memory map

(d) mgmt driver

Step 7: Can the bits in a maintainable build setup

One of the goals of this effort is to create a highly maintainable build system and code base so that minimal effort is needed to keep OS, toolchain and supporting libraries and utilities up to date on a regular basis. One of the key tenets of such a design would be to have all the bits under SCM so that multiple computers and workstations don't have to be updated each and every time a part of the root filesystem or toolchain is modified.

Migrate Bobcat to Linux

Run PMC Sierra cores in 64bit, big endian?

This would require a mod to the Bobcat motherboard. Aiee.

Now that I've got your attention ... by the time we get our stuff ported to big endian, the Bobcat will be but a distant memory in the minds of former employees. The decision as to whether we will run the PMC processors in 64 or 32 bit is a bridge that will have to be crossed when we get to it. Either is doable.

Strategy for upgrading from OpenBSD to Linux (Bobcat)

(1) Immediate strategy for increasing CF size on Bobcat to at least 1GB,

preferably 2GB. See separate doc on CF issues.

(2) Special “to-linux” upgrade program for OpenBSD:

1. Ext3 file system library for OpenBSD.
2. Copy config data and databases from OpenBSD to Linux.
3. Boot Linux CF, and any DB upgrades will be done by Linux version.

(3) Some other strategy entirely?

1. export config data; install from scratch; import config data.

List of work items and mods that might need to be ported

- the upgrade process – will be close to a re-write
- ftp (modified in some way to facilitate upgrade by ftp. in reality, i will probably chuck this in favor of something more secure and robust, like upgrade via ssh using RSA/DSA keys. upgrade could be done on a customer machine directly from a onstor.com server)
- DNS resolver mods (no idea)
- gethostnamadr (almost certainly not needed any more, especially if we use glibc instead of BSD libc)
- port libkvm uses over to use /proc like a normal person
- the upgrade process
- virtual ethernet interface – GPP access to TXRX interfaces (network daemons running on GPP listen on TCP ports on TXRX phys network int)
- rcon
- Please add to this list