Automated BoardFarm: Only Better with Bacon

Christian Höltje
TimeSys Corp.
christian.holtje@timesys.com
Bryan Mills
TimeSys Corp.
bryan.mills@timesys.com

Abstract

In this presentation, we introduce the concept of a BoardFarm, a tool to aid in the development and support of embedded systems. TimeSys had an opportunity to save time and energy that was being spent juggling a limited number of embedded boards among our support staff and developers who are spread throughout the world. We decided to build a system to provide remote access to the boards and to automate many of the tedious tasks such as running tests, booting the boards and installing software including the operating systems, board support packages and toolchains. This allows the developers and support gurus at TimeSys to concentrate on specific problems instead of how each board boots or how a specific board needs to be set up.

We talk about why the BoardFarm was built, how to use it, how it works, and what it's being used for. We also talk about ideas that we have for future improvements. Pigs were harmed in the making of this BoardFarm and were delicious.

1 Intro

Let's talk about embedded system boards. In the fast paced hi-tech world of embedded development, we just call them "boards." These little gizmos are developer prototypes, used to design software that goes in your MP3 player, routers, your new car, and Mars rovers.

These boards usually look like motherboards with the normal ports and such, but not always. Some are huge and others are itsy-bitsy. They all have one thing in common, though. They are expensive. These boards are usually bleeding edge technology and are only made available so that developers can get software out the door before the boards become obsolete. This means that usually a company can only afford one (or maybe two) of a board that they are working on.

The BoardFarm is an online tool that gives users remote access to all the boards. It is an interface to the boards and includes automation and testing. The goal is make the boards available to all users, everywhere, all the time.

2 Problems

Fred and Sue have problems.

The first problem is finding a board. “Fred had that last week,” but Fred says that Sue has it now, but she’s out of the office. Where *is* that board? This is asset management, but it is complicated by having the boards move between developers and with no central repository. It is even more complicated if your coworkers are not in the building or worse, in a different town or state! “Oh, yeah, Sue took that with her to California, did you need that?”

What if Fred and Sue need the same board at the same time? “Board snatching” is the common name for a form of stealing (not usually investigated by Scotland Yard) and is quite common when two people have deadlines involving the same board. If they’re lucky, they can cooperate and maybe work at different hours to share the board.

Finally, and possibly the most frustrating, is having to rediscover how to boot and configure a board. “Didn’t Fred work on that board a year ago?” A lot of the boards are unlabeled, only configurable through arcane methods or require sacrifices (under a full moon) to enter the boot loader. Sure, the developer should write it down. Fred usually does. But writing down a process is never as good as having functioning code that does the process. And the code is well exercised which means you know it works.

3 The BoardFarm: Grab a hoe and get to it

So what do we do? Panic? Sure! Run around in circles? Why not? Write code? Nah... No, wait, that was right answer.

So now, instead of investing in sharp cutlery, Fred and Sue can use our BoardFarm. The boards are safely locked away where they will no longer be stolen or used as blunt weapons, yet are easily accessible and even more useful than before.

What was our secret? We used the awesome power of python and bacon to create the BoardFarm. The BoardFarm acts as resource manager, automating some tasks and scheduling when Fred and Sue can use boards. It knows how to configure the board, install a board support package (BSP), boot the board using the BSP, and even how to run tests. This means that Fred and Sue can worry about their own problems, not who has the board or how the thing boots.

3.1 A day in the life of Fred and Sue

Sue has been working on a really tough spurious interrupt problem in some kernel driver. Fred has been running various tests in the PPC7xx compiler, using the same board as Sue because someone discovered a bug.

So Sue sits down at her desk to start a BoardFarm session. She uses ssh to connect to one of the BoardFarm testroot servers. Once there, she reserves a testroot and specifies the BSP that she is going to use. The BoardFarm creates a clean testroot, a virtual machine, and installs the selected BSP. This virtual machine is called the testroot. She logs into the testroot and starts her work.

Since the BSP is already installed and the BSP has the kernel sources and proper cross compilers, she is ready to start working on her problem. As she progresses, she compiles the kernel then tells the BoardFarm to run a portion of a testsuite which has been really good at triggering the system-crippling bug. The BoardFarm

grabs an available board, boots the board using her *new* kernel, compiles the tests in the testroot and then runs the tests on the board. She checks the results and keeps on working on her bug. She can repeat these steps as often as she wants.

Fred is working his way through the test-results that show his bug. He got the test results from an automated test that ran last night. The automated tests run every time a change happens to a major component of the BSP. Last night, it detected problems with the toolchain. Since Fred broke it, he gets to fix it.

Fred sits at his desk. Instead of requesting a testroot to work in, Fred uses his own workstation, preferring to submit test requests via the web interface. Using the test results, he figures out where the problem probably is and starts working on his bug. After an hour or two, he has what he thinks is a good rough fix. He compiles the toolchain and submits it to the BoardFarm web page to automatically test. He tells the BoardFarm to use specific testsuites that are good for testing compilers. The BoardFarm reserves the appropriate board when it's available, sets up a testroot, installs a BSP and his *new* toolchain, boots the board, compiles the tests and then runs them. When the tests finish, the BoardFarm saves the results, destroys the testroot and unreserves the board. Fred uses the results from the tests to analyze his work.

This works even if Fred and Sue are using the same board. If Fred and Sue submit requests that need the board at the same time, then the BoardFarm schedules them and runs them one after another. Either Fred or Sue's results might take a little longer, but neither has to hover around the desk of the other, waiting to snatch the board. Instead, if their jobs conflict, they can just go grab a bacon sandwich from the company vending machine while the BoardFarm does the needful.

By the end of the day, Fred feels confident that he has throughly squished the PowerPC bug. To be sure, he should really check his new and improved toolchain out on more than just the one board. So he kicks off full test suite runs, using his new toolchain, on all the PPC7xx boards and then goes home. While each test suite can take 8 hours or more, they can all run in parallel, automatically, while Fred relaxes in a lawn chair with his Bermuda shorts and a good Belgian beer, out front of his south-side Pittsburgh town home.

We'd like to take this opportunity to point out some interesting facts. Neither Fred nor Sue had to worry about booting the boards or installing the BSPs. The BoardFarm knew how to boot the boards, install the BSPs and did it without human primate intervention. This is a real time-saver as well as a great way to prevent Sue and especially Fred from getting distracted.

4 Architecture: The design, layout, blueprint, and plan

How is the BoardFarm put together? The BoardFarm is made up of four parts: a web server, a database, testroot systems, and the boards themselves. There is also some other support hardware: an SMNP-controlled power strip, a network switch, and a serial terminal server.

The web server is running Apache2 [2]. The web pages are powered by PSE [7] which uses `mod_python` [1] to get the job done. We chose this combination to allow fast development and to harness the power of Python's [6] extensive library.

The database system is powered by PostgreSQL [5]. It's otherwise just a normal humble box. The database itself isn't that radical.

It's mainly used to track who is using a board and what architecture and model a board is. We need this information to match a given BSP to one or more boards.

The testroot systems are normal boxes running Python and have a standardized OS image that is used to build a testroot. The OS image must have the components needed to compile, remotely test and debug problems on the boards. The testroot is an isolated environment that could be implemented as a chroot or UML [8] environment.

The boards themselves are all different. This is one of the hard parts of dealing with embedded developer boards. Some of them even require special hardware tricks just to power on.

All boards are powered via a network-controlled power strip. This allows us to reboot a system remotely. Usually. Some boards require a button to be pressed which requires interesting solutions ranging from relays to LEGO[3] Mindstorms[4].

A serial terminal server is needed to connect all the serial ports on the boards to the test servers. Since the serial terminal servers are networked, we can share one board among many testroot servers.

5 Show me the BoardFarm

Let's talk about the user interface. The BoardFarm has two user interfaces: a web page and a remote shell.

The web page is the easiest to use. Fred likes it because it's simple, direct, and has pretty colors. To run a test, Fred chooses the BSP, picks some tests and then clicks the "Test" button. After the BoardFarm is done, Fred gets an email with a link to the web page with all the test results.

Sue preferred the more powerful interface: the testroot command line shell. The shell is connected to via ssh and looks remarkably like a bash shell prompt, mainly because it is. Within the testroot, Sue has a series of BoardFarm shell commands.

These shell commands are very powerful. They provide control of the testroot, the BSPs, kernel, and the board itself. Using these commands, Sue can clean out the testroot, reinstall the BSP, choose a custom kernel, reserve a board, boot the board, power-cycle the board, run tests on the board, connect to the board's serial console, unreserve the board and save the test results. Furthermore, since these are normal shell commands, Sue can write custom batch scripts.

6 Successes so far

This system is actually a **fully** operational death star… It has been running now for over 6 months inside TimeSys. The BoardFarm is used on a daily basis. TimeSys and the BoardFarm are located in Pittsburgh, Pennsylvania, and we have remote users in California and even India.

We have used the BoardFarm to successfully troubleshoot problems, help develop new BSPs and to test old BSPs. The support team uses the BoardFarm to reproduce and troubleshoot customer problems. The documentation team uses the BoardFarm to confirm their manuals and to gather screen shots.

In addition to manual tasks, the automated testing is used with various TimeSys test suites to test BSPs on demand (just one click!^{TM}). In

addition, new BSPs are automatically tested as soon as they are successfully built in our build system.

It is only fair to point out that the BoardFarm is actually a part of TimeSys's in-house build environment. This integration makes manual usage easy and provides the starting point for the automated testing. As the build system successfully finishes a build, the BoardFarm queues the build to run tests when the appropriate boards are available. This makes the "change, compile, test" cycle much shorter.

7 Known Limitations

We knew from the beginning that we couldn't have everything (at least not right away). The original plan for the BoardFarm was to only provide an automated testing environment. Since then we have added the ability to do actual development using the BoardFarm. Since these goals have evolved, we have run into some limitations with our original design.

In general though, these limitations boil down to one real problem. These are developer boards, not meant to be production level devices, that at times require someone to actually go and visit the board. For example, a board might need a button pressed to power on. Or certain error situations can only be diagnosed by looking at a pattern of LEDs.

Another aspect of the "no access" problem is developing peripheral device support. To troubleshoot USB, you need to be able to plug-in and remove devices. To check that PCMCIA is working, you have to try various classes of devices. And so on.

The only other limitation isn't a technical problem, it's a social one. Developers are the ultimate power users. Most developers hate having something between them and what they're working on. Some developers appreciate the advantages of having the BoardFarm help them. Others try to work around the BoardFarm however they can. And a few of the extremists just demand that the board be handed over to them.

8 The future's so bright, we gotta wear shades

Like most projects we have grandiose plans for the future. We have plans to make the BoardFarm do test analysis, boot boards that require a button press, and integrate with our project management tools.

The BoardFarm collects all test results but it doesn't understand them. Some tests are more important than others and sometimes multiple failures can be due to just one root problem. The BoardFarm can't make this distinction. We would like the BoardFarm to help us understand our trends in failures and help recognize where we need to focus our efforts.

Some boards require a button to be pressed. Despite experimentation with electrodes and lawyers, we haven't found a sufficiently reliable solution. Plus, the lawyers were expensive. We have our crack electrical engineering team working on a solution.

The BoardFarm is just a piece of a larger system. Even though it works with the large system, it isn't feeding back information into all the systems. Ideally, we'd like it to file bugs and test things that have we have bugs open for.

9 Conclusion

In conclusion, we only have this to say: “Bacon Rocks.”

Good Night!

References

[1] Apache python module. `http://www.modpython.org/`.

[2] Apache web server. `http://httpd.apache.org/`.

[3] Lego. `http://www.lego.com/`.

[4] Lego mindstorms. `http://www.legomindstorms.com/`.

[5] Postgres sql database. `http://www.postgres.org/`.

[6] Python language. `http://www.python.org/`.

[7] Python servlet engine. `http://nick.borko.net/pse/`.

[8] User mode linux. `http://user-mode-linux.sourceforge.net/`.

Proceedings of the Linux Symposium

Volume One

July 20nd–23th, 2005
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*
Stephanie Donovan, *Linux Symposium*

Review Committee

Gerrit Huizenga, *IBM*
Matthew Wilcox, *HP*
Dirk Hohndel, *Intel*
Val Henson, *Sun Microsystems*
Jamal Hadi Salimi, *Znyx*
Matt Domsch, *Dell*
Andrew Hutton, *Steamballoon, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*