ELC 2010

Effective Scripting in Embedded Devices

Steve Bennett

WorkWare Systems

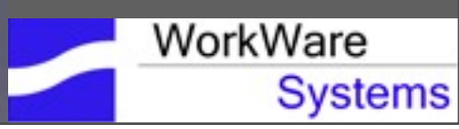
What is Embedded?

ELC 2010

WorkWare Systems

Creating an Embedded Product

- Time to market
- Quality
- Features
- Cost
- Size
- Performance
- Linux kernel
- uClibc
- Busybox
- Other open source
- Custom drivers
- Custom applications



Embedded Applications

Embedded Minimalists

- linked list
- hash table
- exec wrapper
- config parser
- customisation API

Application Porters

- C++ toolchain
- Boost
- PostgreSQL
- byte order
- unaligned access

Greenspun's Tenth Rule

Any sufficiently complicated C program contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of a scripting language.

Language Strengths

C/C++

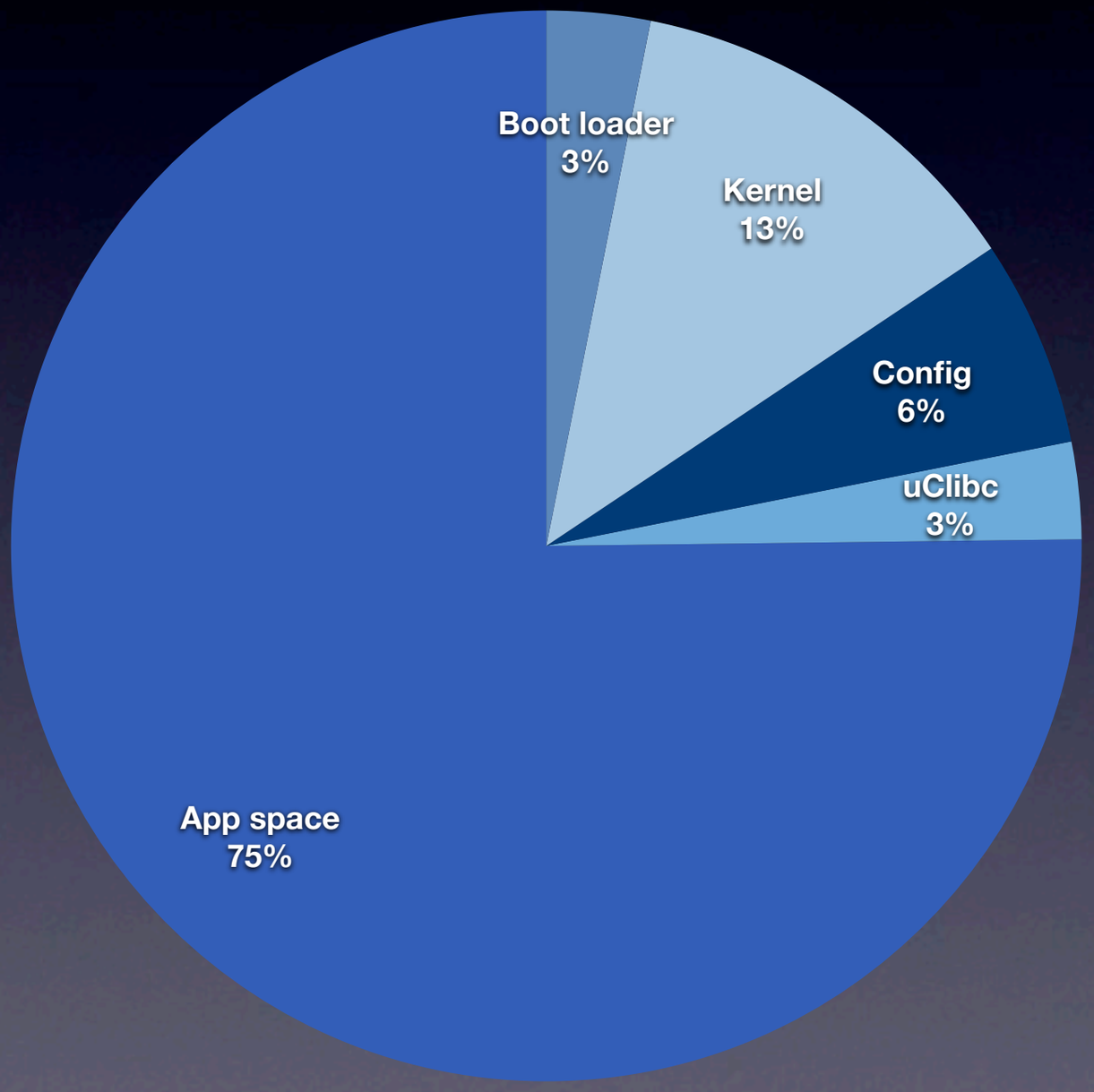
- Bit/byte twiddling
- Efficient storage
- Access entire system API
- Compiled code

Scripting

- String mangling
- Lists, Dictionaries
- Searching, Sorting
- Customisation

Make it Fit

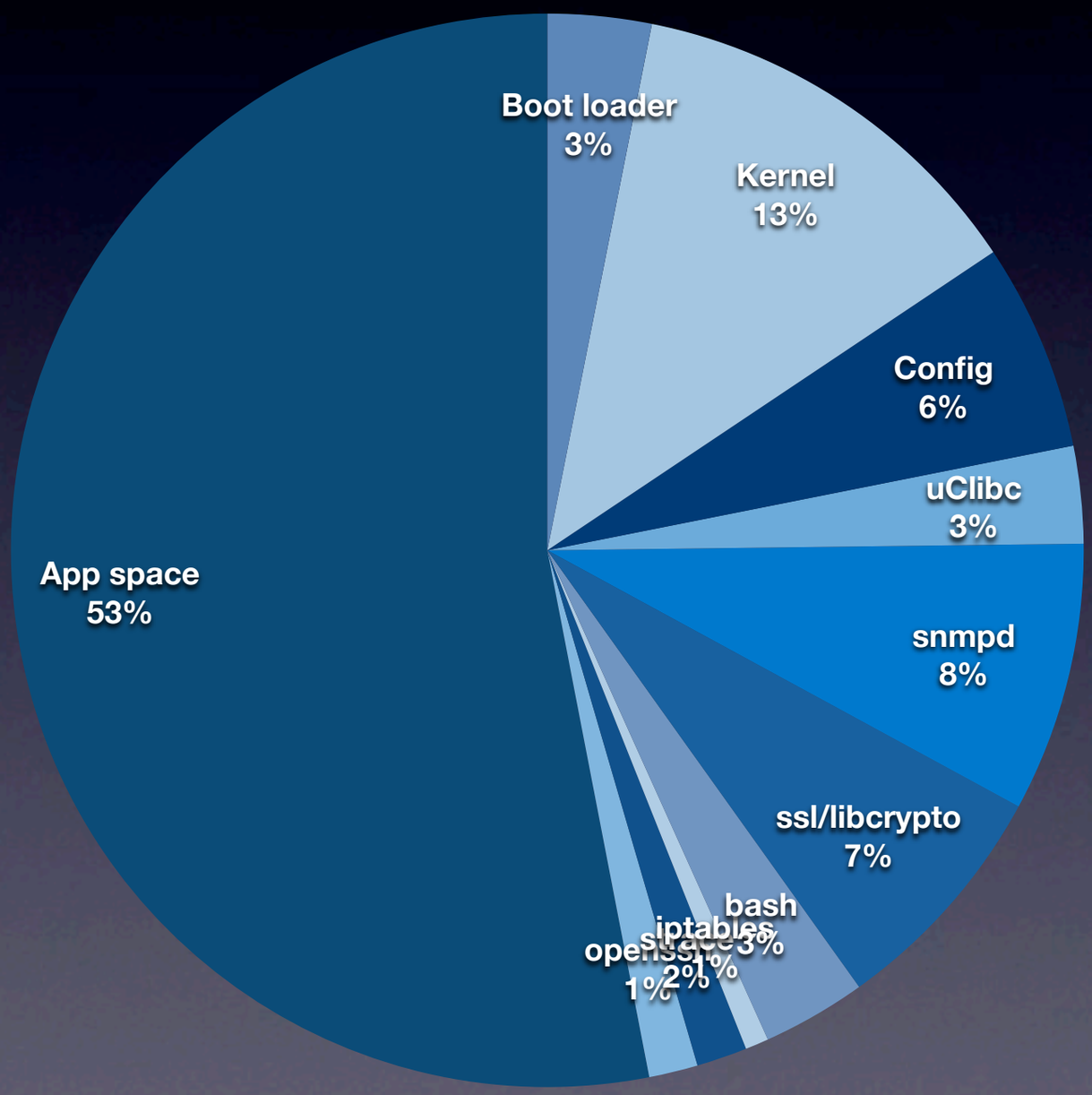
Percentage of 8MB NOR flash used



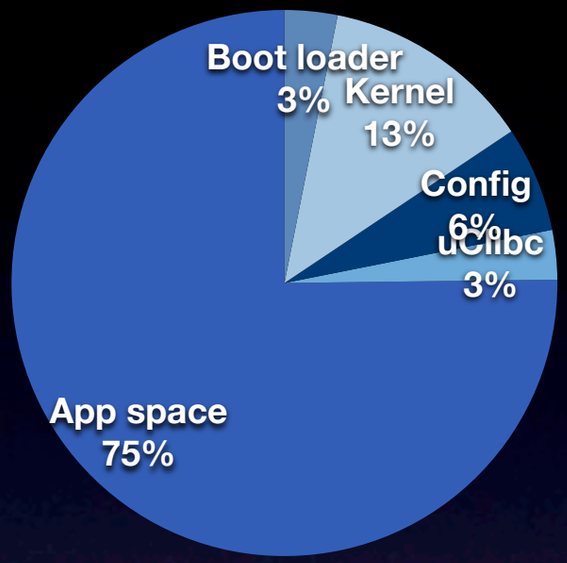
ELC 2010

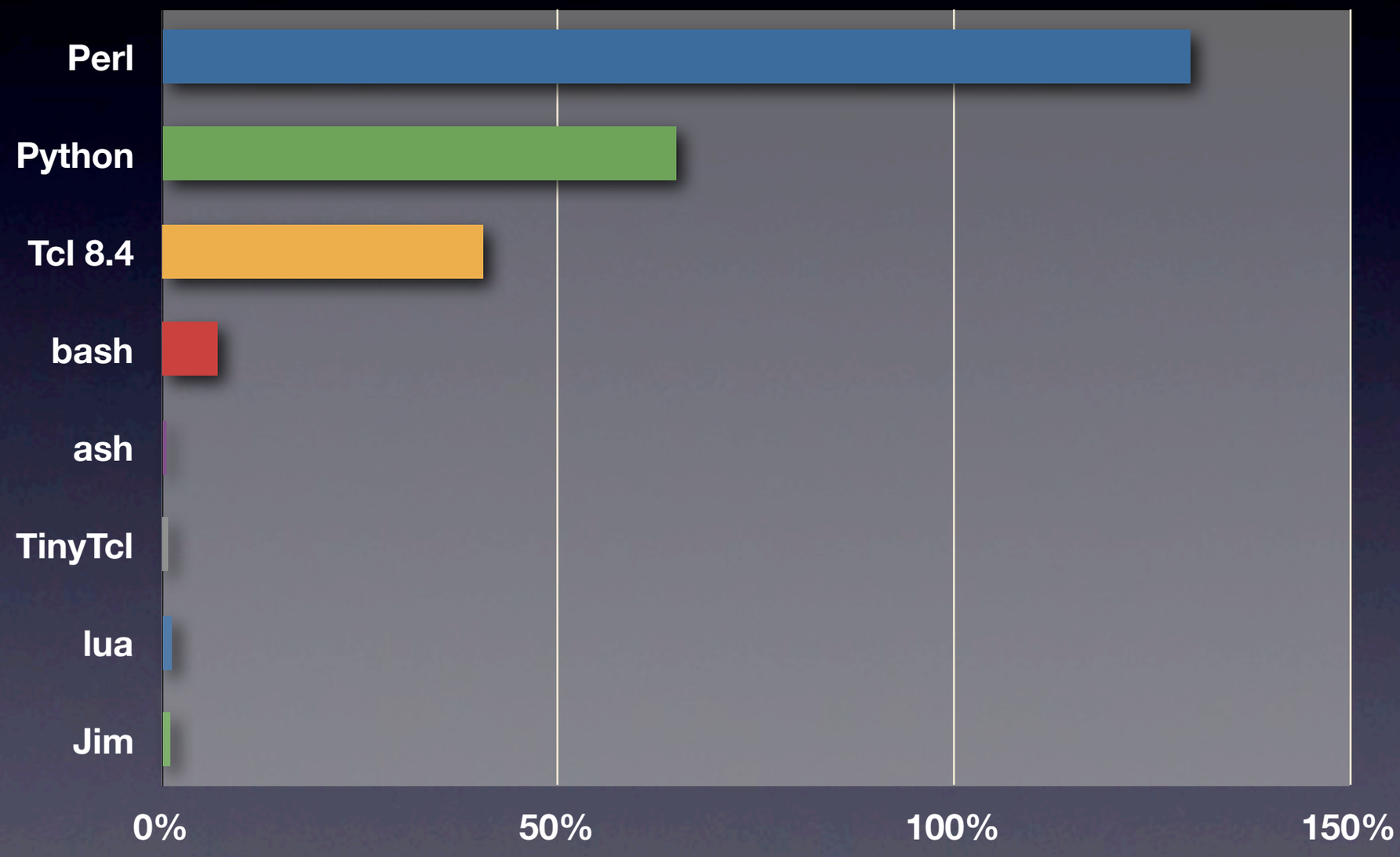
WorkWare Systems

Make it work


Percentage of 8MB NOR flash used
Boot loader
3%
Kernel
13%
Config
6%
uClibc
3%
snmpd
8%
ssl/libcrypto
7%
bash
3%
iptables
2%
openssl
1%
App space
53%

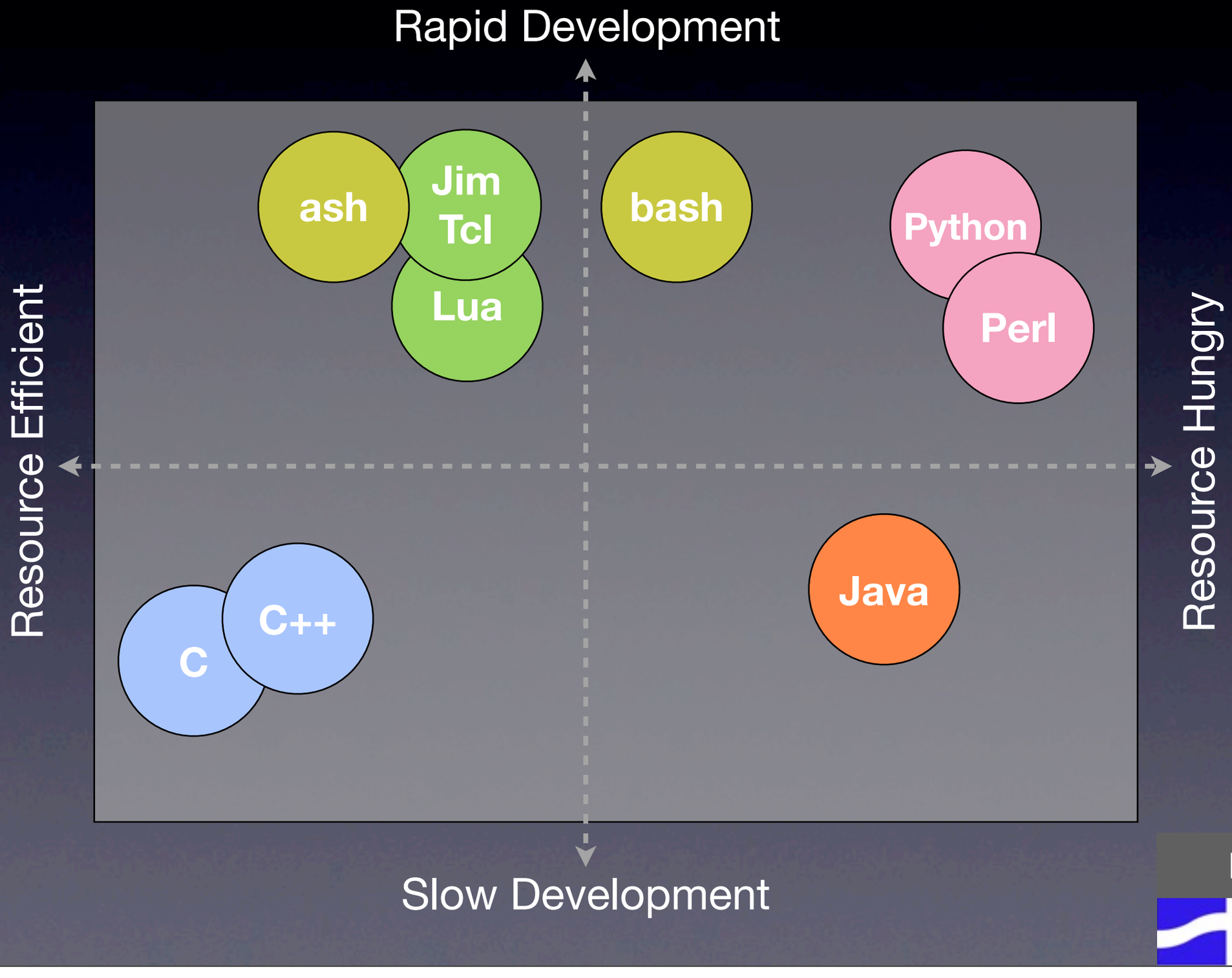
Add Scripting





Percentage of available space used by “core” scripting language

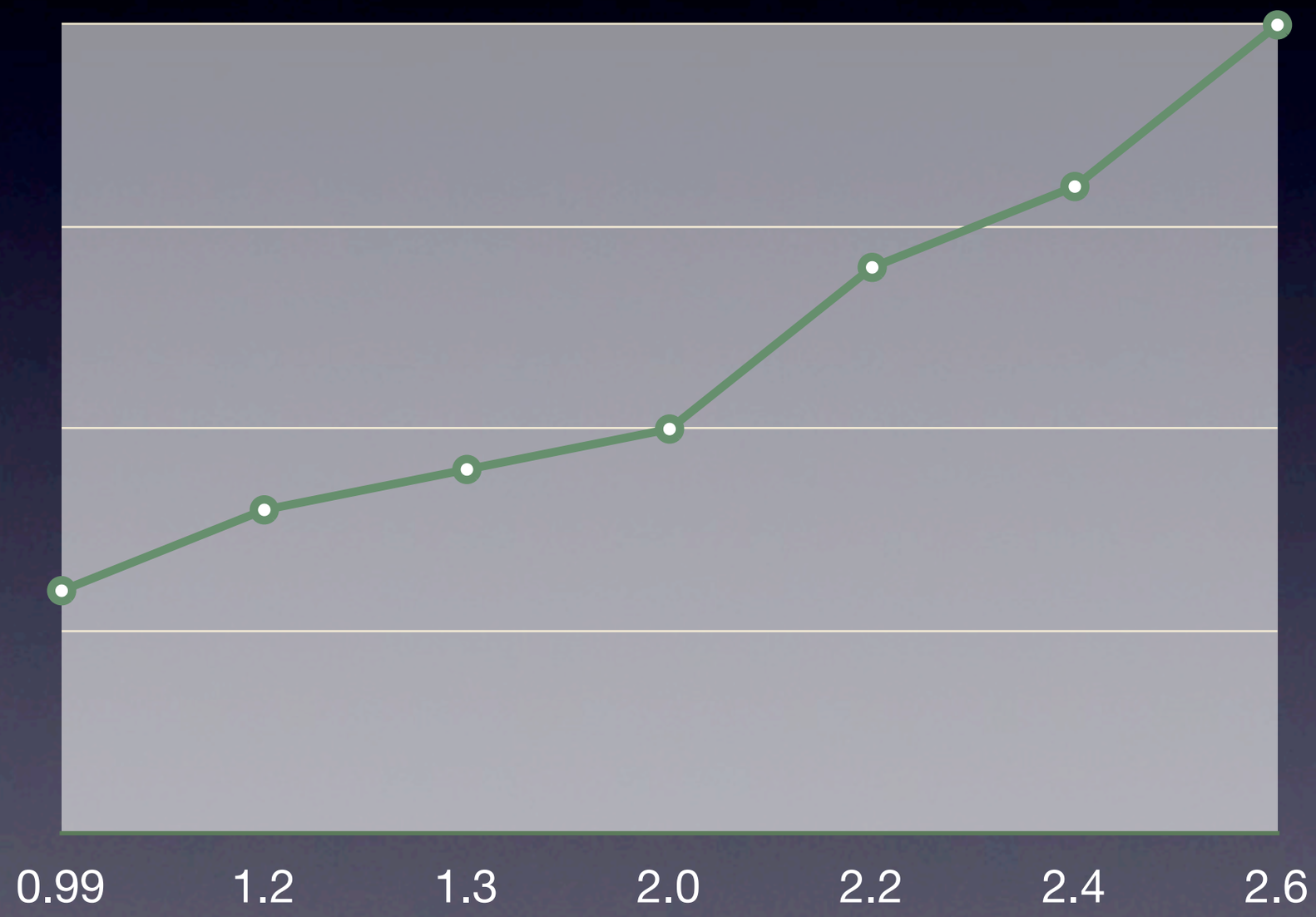
Languages Attributes



ELC 2010

WorkWare Systems

Growth over Time



Note: Sizes are indicative only

Making big things small

- It is hard since all features are critical to *someone*
  - Minimal Tcl - 5+ years with no progress
  - Deeply Embedded Python - abandoned
  - miniperl - unsupported
- Much easier to start small and focussed

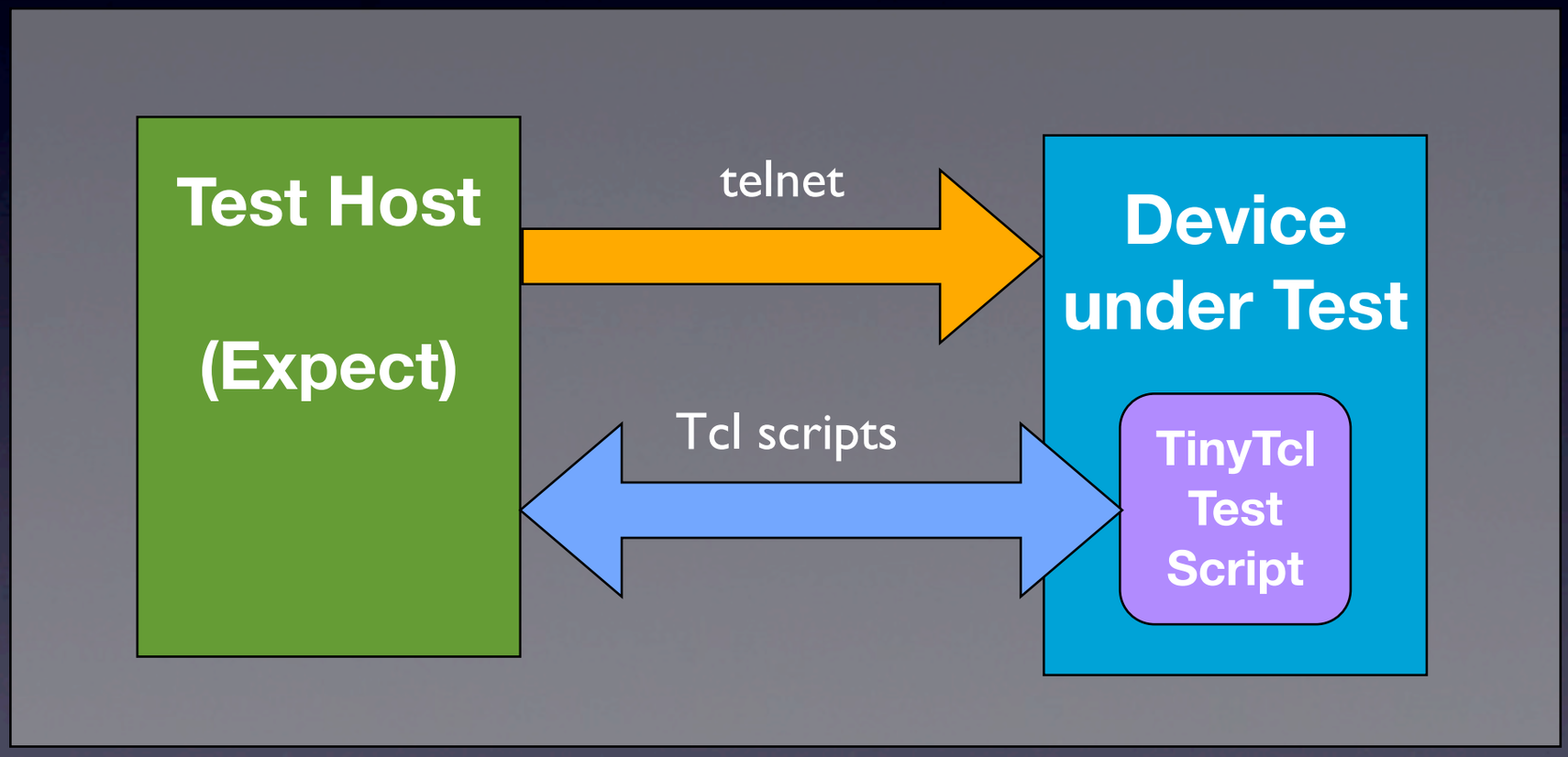
Size - Speed

All things being equal, large applications and libraries are slower to load and run than small applications and libraries

System	Time	System Calls	Relocations
Intel(R) Core(TM)2 Quad CPU 2.33GHz, 4GB RAM **Tcl 8.4 (glibc)**	43ms	173	3740
XScale-IXP42x (v5b) 266MHz, 32MB RAM **Jim Tcl (uClibc)**	1ms	37	766

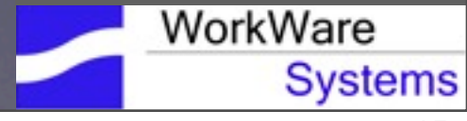
Simple 'Hello World' Test

Case Study
Automated Testing



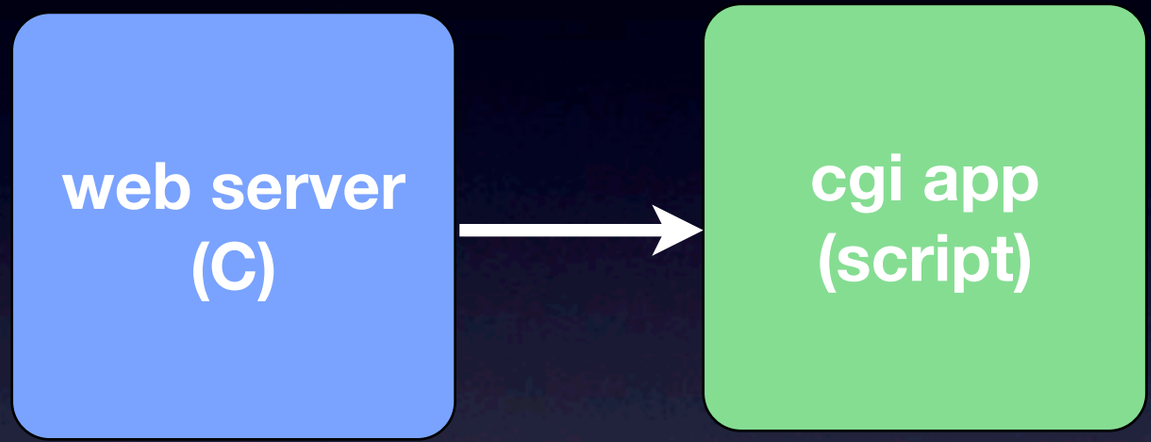
Expect + inetd + TinyTcl

```
source $testlib
use netconf net
test cable {
        # Find a dhcp connection we can use
        array set conn [netconf_find dhcp]
        # Configure it
        remote dev=$conn(dev) devname=$conn(devname) {
                config load -update
                set eth [config ref eth<devname=$dev>]
                set o [config new dhcp interface $eth]
                config set $o type cable
                if {$devname != "eth0"} {
                        config set $o fwclass wan
                }
                config set $eth conn $o
                config save
        }
        # Wait for it to come up
        net_wait $conn(intf)

        pass "cable connection on $conn(intf) OK"
}
```

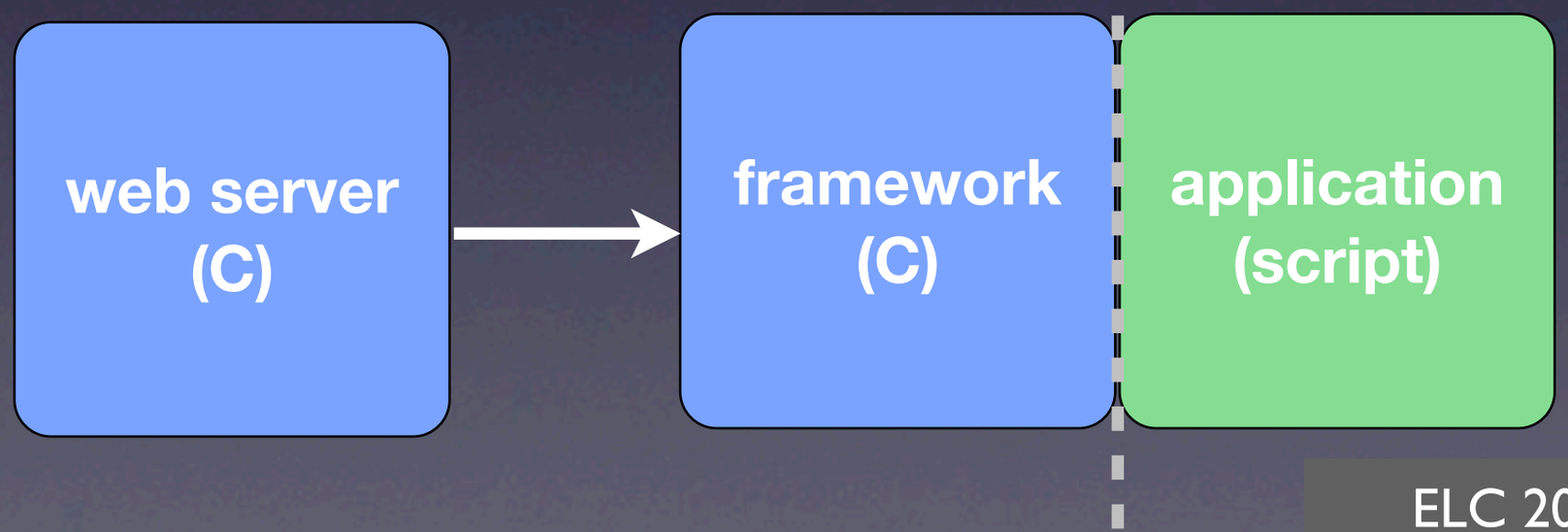



Case Study
Web Framework

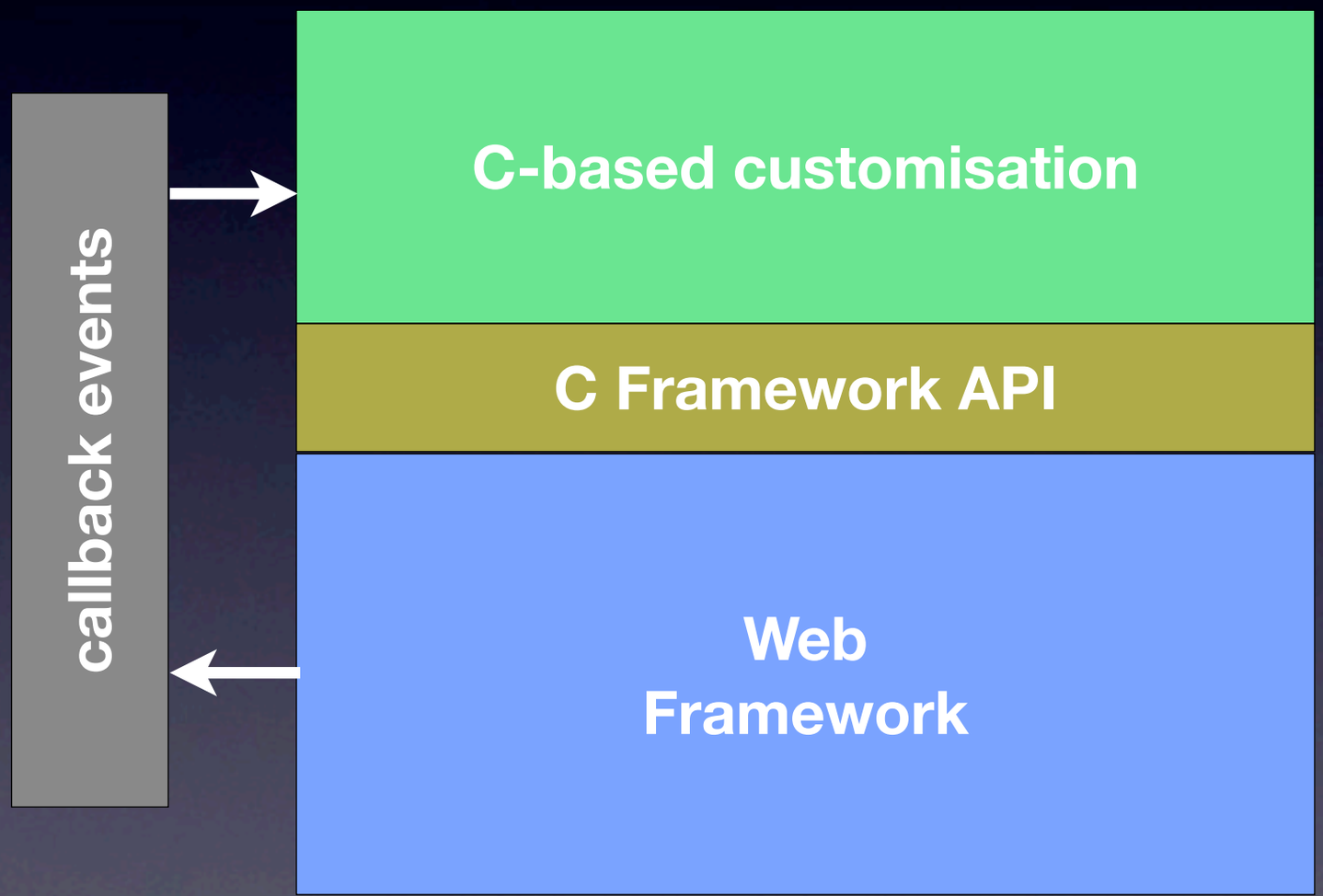
Traditional



Embedded Scripting

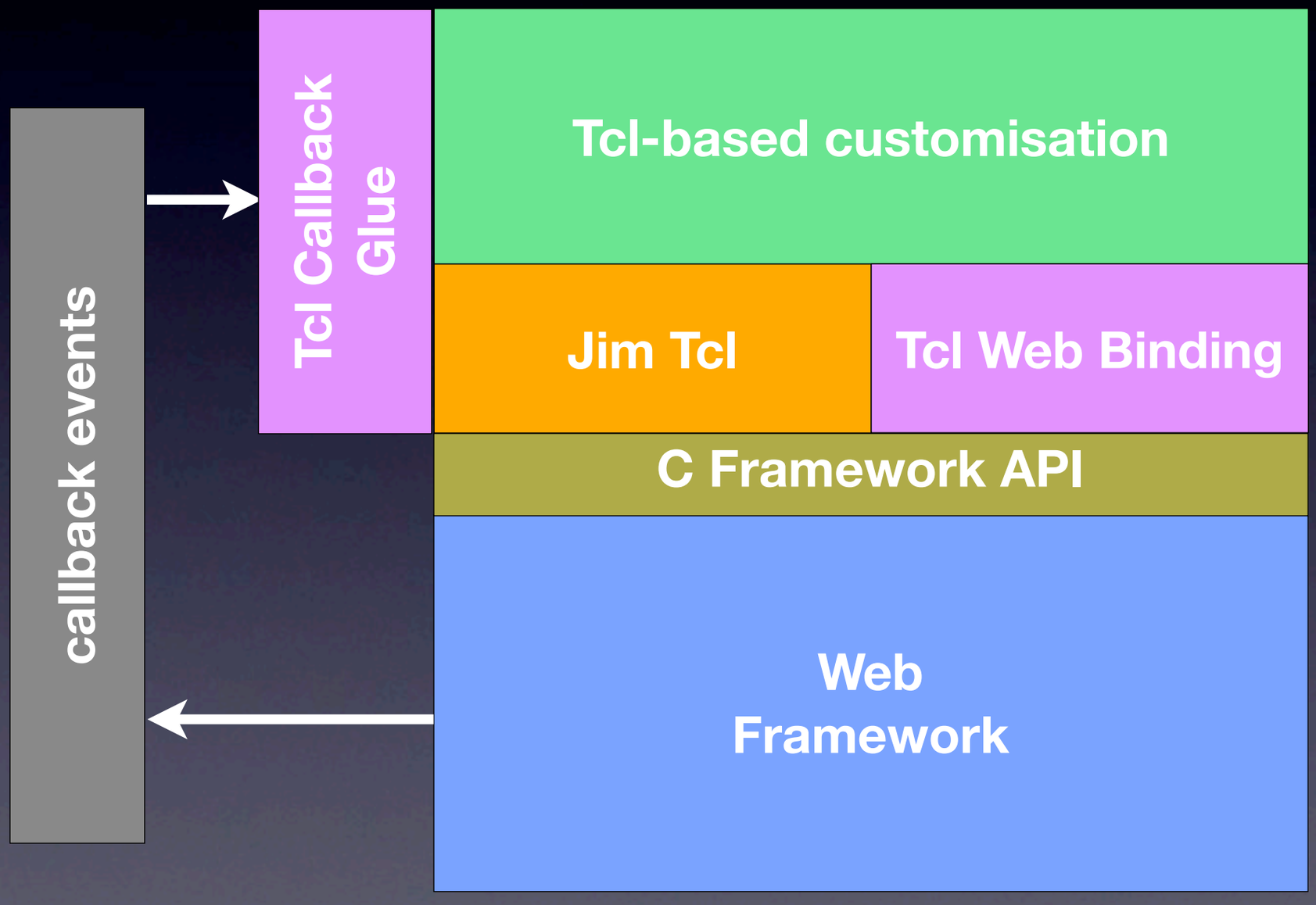


µWeb
C-based Customisation



```
submit -c {
        const char *tz = cgi_get("tz");
        /* find timezone spec for selected TZ */
        FILE *fh = fopen(ZONEFILE, "r");
        while ((fgets(buf, sizeof(buf), fh) != NULL) {
                /* parse line,
                 * match timezone,
                 * write to /etc/TZ
                 */
                ...
        }
        fclose(fh);

        /* write ntpserver */
        snprintf(buf, "%s/ntpserver", cgi_configdir());
        fh = fopen(buf, "w");
        fprintf(fh, "%s\n", cgi_get("ntpserver");
        fclose(fh);

        /* should use msntp.pid, ...*/
        system("killall msntp");
}
```

µWeb
Jim Tcl Scripting



ELC 2010

WorkWare Systems

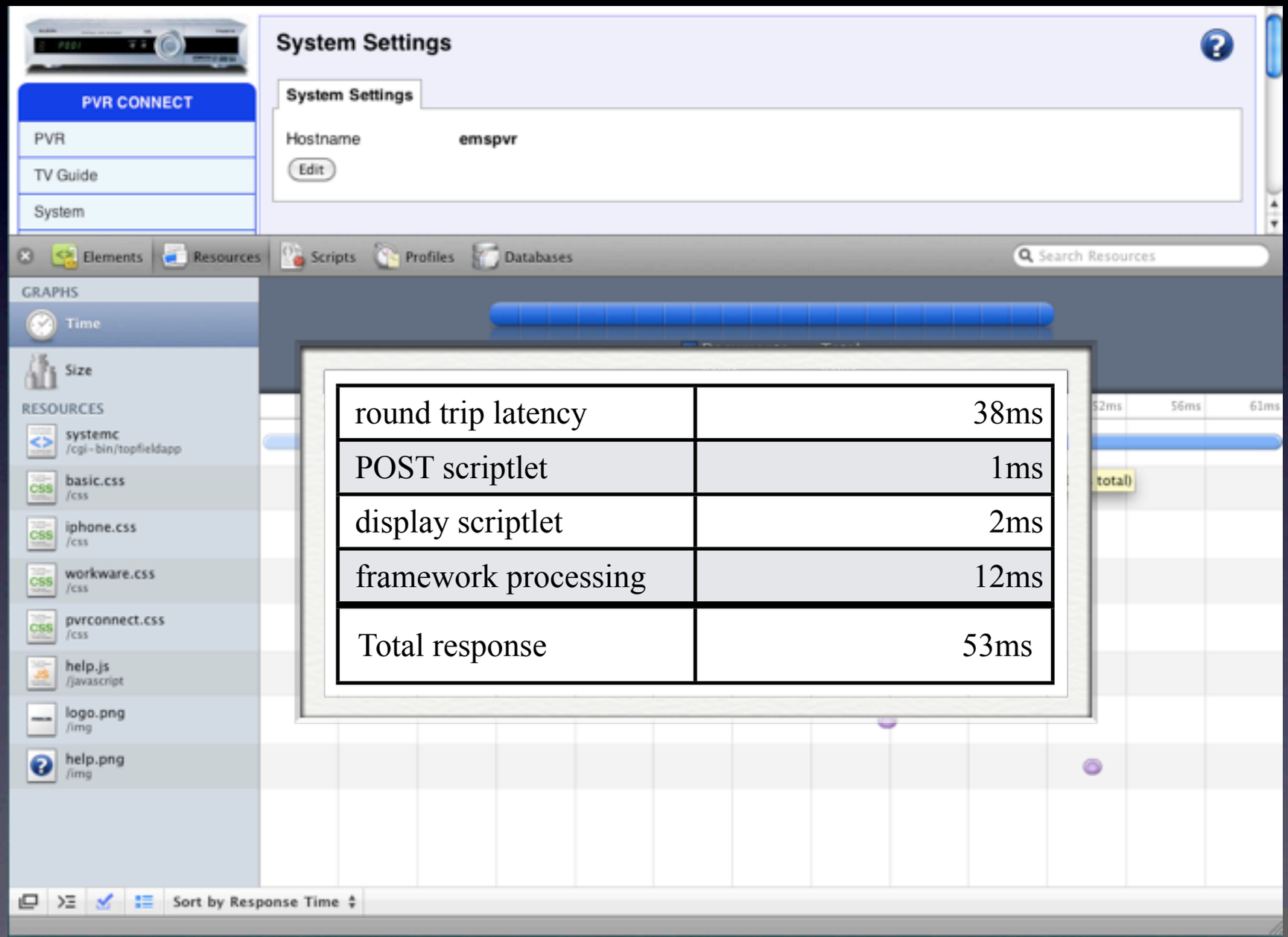
```
submit -tcl {
        # read timezones
        set zones [readfile $ZONEFILE]

        # write /etc/TZ
        writefile /etc/TZ $zones([cgi get tz])

        # write ntpserver
        writefile $CONFDIR/ntpserver [cgi get ntpserver]

        # kill (and respawn) msntp
        kill -TERM [readfile /var/run/msntp.pid]
}
```

What scriptlets do

- Access application API (Tcl commands)
- Examine/update strings, lists, arrays
- Use standard Tcl commands
- Interact with OS - files, commands, processes

How Fast is it?

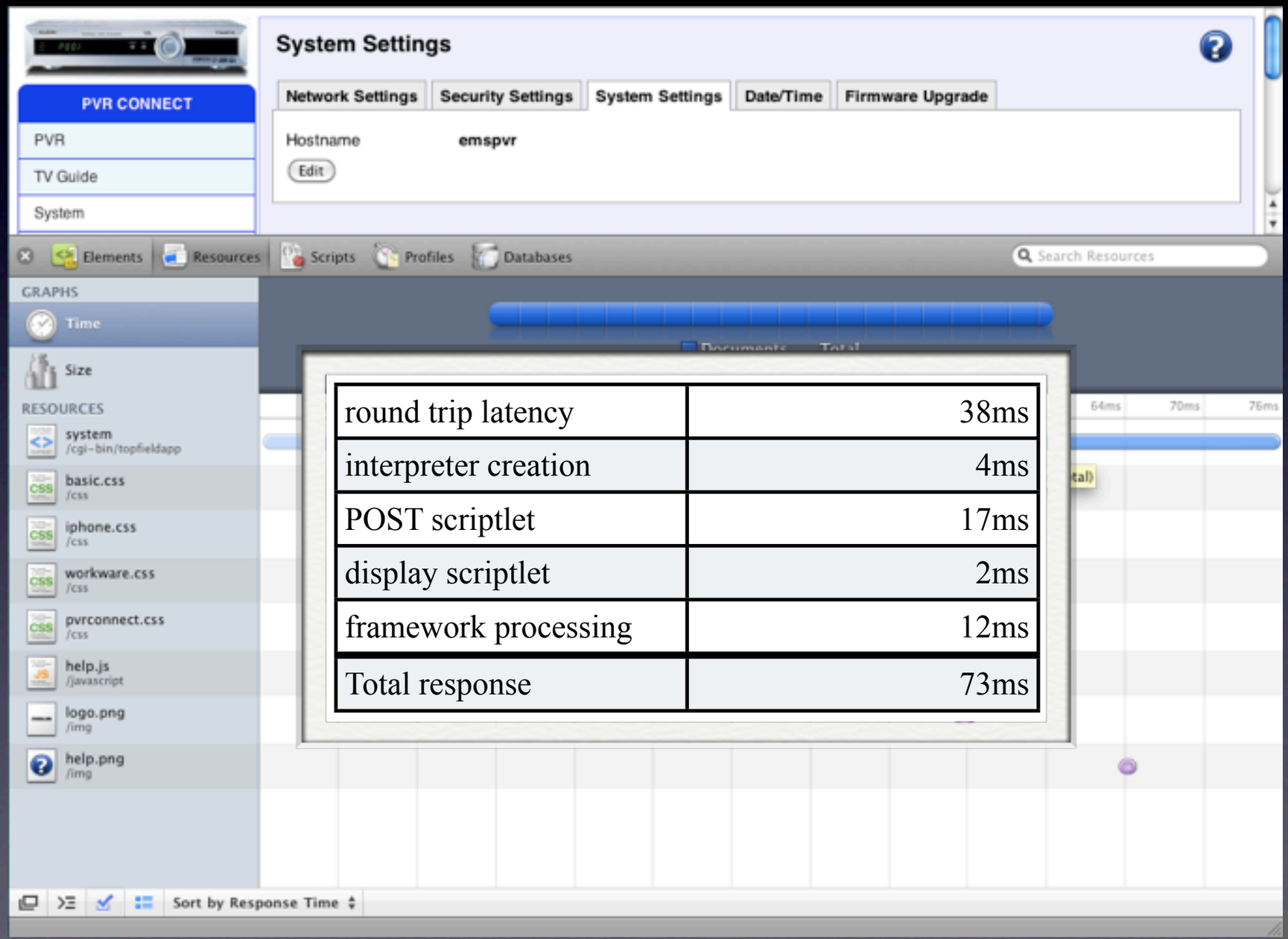
C-based



round trip latency	38ms
POST scriptlet	1ms
display scriptlet	2ms
framework processing	12ms
Total response	53ms

How Fast is it?

Tcl Scripting



round trip latency	38ms
interpreter creation	4ms
POST scriptlet	17ms
display scriptlet	2ms
framework processing	12ms
Total response	73ms

Timing Comparison

	C-based	Tcl-Based
round trip latency	38ms	38ms
interpreter creation	-	4ms
POST scriptlet	1ms	17ms
display scriptlet	2ms	2ms
framework processing	12ms	12ms
Total response	53ms	73ms

Possible Applications

- control cameras, frame rate, image processing, network access
- environmental sensor data gathering, analysis
- industrial control

Scripting Language Requirements

- Written in portable C
- Designed to be embedded, not standalone
- Small
- Fast to start
- Modular, to allow unneeded features to be removed
- BSD or equivalent licence

- 10x speed of TinyTcl, 50% of Tcl 8.4
- Small (80-150KB)
- Designed for embedding
- BSD licence

Jim Tcl Features

- regular expressions
- exec
- associative arrays, lists
- file, glob, open, close, read, write
- functional programming
- accurate error reporting
- arrays as first class objects
- 64 bit integers
- strings containing nulls
- list expansion operator
- simple packages
- event loop, sockets

Lua

- Designed for embedding
- Portable
- Small
- Byte code
- BSD licence
- Used in World of Warcraft

Other Scripting Languages

- Pawn (formerly Small)
- Pike
- Nesla

Leveraging Scripting
Ad-hoc scripts

```
Vendor/product Version 1.0  Mar 19 12:23:35 EST 2010

  1.  Modem 1                              [Active]
  2.  Modem 2                              [Not Installed]
  k.  Modulation Control                   [Running]
  t.  Modem Test Signal (0x1B)             [None (0)]
  m.  Modulation (0x01)                    [QPSK (0x00)]
  q.  Quit

Select option []:
```

Simple Menuing System for Internal Use

Leveraging Scripting
Prototyping

- Fills the gap between shell scripts and C
- Small daemons
- Configure systems
- Exec commands
- Parse files
- Reload config on SIGHUP

Leveraging Scripting
Replace Complex Shell Scripts

- String and data structure manipulation
- Invocations of sed/awk/grep are slow
- Shell quoting hell
- No floating point math
- Start-up time may be critical

Leveraging Scripting
"Free" CLI

- Easy to add Command Line Interface
  - User Interaction
  - Debugging
- Possible mechanisms:
  - Unix domain sockets
  - Special startup mode

Pitfalls

- Excessive stack usage
- Unicode support
- No-MMU support
- Licencing
- IP Leaking

More about Jim Tcl

- Expand operator
- List-dictionary duality
- Source location tracking
- Get it:
  - http://jim.workware.net.au/