

# RADTools

## Towards Automated Management for Distributed Applications

CS294-1 RADS – Fall '06  
Greg Gibeling  
Lilia Gutnik  
Nathan Burkhart

1 12/14/2006 RADTools

## Overview

- Problem
  - Requirements
  - Solution: RADTools
- Internals
  - RADServices
  - Monitoring
  - RCF Libraries
- Conclusion
  - Obstacles
  - Report card
  - Results
  - Future Work
- Questions & Demo!

2 12/14/2006 RADTools

## Problem

- Individuals building large services
  - Lots of moving parts
    - Webserver, dispatchers, database, cache, VMs, etc...
  - Configuration is very difficult
    - Class labs are a good example
      - 1-4 person groups with VERY good instructions still had trouble
      - Very hard to automate (e.g. with SML)
- Available tools
  - Capistrano: Automated deployment; no abstraction at all
  - Scyld: Vapourware?
  - Various linux config tools: dead projects, not distributed
  - No one tool to cover it all

3 12/14/2006 RADTools

## Requirements

- Simple
  - Provide an abstraction a non-expert can work with
  - Duplicate the CS294-1 labs in minutes not weeks
  - "So easy a professor could use it"
  - Interface with SML code
- Uniform
  - Expose buried config files and options
  - Provide a uniform way to control (start/stop) services
  - Simplify the documentation
    - Documentation is scattered at best
    - With a uniform interface there is a lot less to document

4 12/14/2006 RADTools

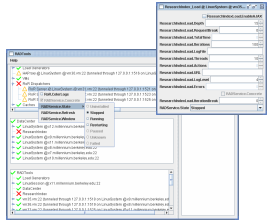
## Solution: RADTools (1)

- Centralized Control
  - Goal of RADLab is to allow ONE person to build a huge service
  - Simple GUI: Status & Commands in a single window
  - Monitor statistics
    - Generate basic graphs
    - Reads from nagios (already widely used)
- Component Based System
  - Service is the basic element
  - Independent component structure models
  - Easily extendable
    - Add components, GUI is automatically generated
    - Adding a new service, or a new config option to a service is trivial

5 12/14/2006 RADTools

## Solution: RADTools (2)

- Simple GUI
  - Start/stop/restart
  - Modify config parameters
- Failure management
  - Failure detection & propagation
  - Failed is a normal state of operation
- Automation
  - 5 lines of code can boot a VM, run dispatchers and configure HAProxy



6 12/14/2006 RADTools

## Internals: RADServices (1)

- The base unit of composition & management
  - Has a uniform state representation
  - Uses RCF Component Framework
- Independent component structure models
  - Composition: Composition and abstraction of services
  - Dependency: Physical dependencies (e.g. VM on PM)
  - Management: Requirements for being able to manage a service
  - Communication: Will model communication & service request paths

7 12/14/2006 RADTools

## Internals: RADServices (2)

- Expansion
  - Adding a new config option: 3 lines of code
    - Declarative programming through java annotations makes this a breeze
  - Adding a new RADService: 2-4 hours, including debugging
  - GUIs will be automatically generated
  - Creating a new top level service (Instead of ResearchIndex)
    - One object instantiation per RADService
  - Can add any one feature in less than an afternoon

8 12/14/2006 RADTools

## Internals: Monitoring

- Integrated with Nagios
  - Widely used already
  - Easily adapted for any Nagios plugin
  - A little tricky to extract useful numbers
- Graphs
  - Using Excel is really a bad plan
    - No faith in graphs
    - Painful
  - Monitor stats dynamically over time
  - Line graph, color chart

9 12/14/2006 RADTools

## Internals: RCF Libraries (1)

- Event Model
  - EventSources generate events, EventSinks receive them
  - No synchronization or threading
  - Allows failure management
    - Failures are propagated down the management & dependency trees
    - Failures are propagated up the composition tree
  - The basis of policy decisions

10 12/14/2006 RADTools

## Internals: RCF Libraries (2)

- Transactional Data Structures
  - All data structures can syndicate events on mutation
    - Overhead is low (1 test) if there are no registered sinks
  - Supports A&C of ACID
    - Transactions are reported atomically, and can be nested
    - I: Can be added by using a synchronizing wrapper
    - D: Could be added with logging
  - Allows changes to e.g. HAProxy pools to trigger config update & restart

11 12/14/2006 RADTools

## Internals: RCF Libraries (3)

- Component Framework
  - A user level reflection framework
    - Builds on java reflection where possible
    - Similar to JMX
    - Methods -> Operations
    - Fields -> Properties
  - Allows for dynamic reflection
    - Including non-local declarations
    - Nagios data becomes properties
  - AutoGUI is generated from component information
  - Supports static usage, through java annotations
    - Annotate a field to make it a property

12 12/14/2006 RADTools

## Conclusion: Obstacles

- SSH Library
  - No documentation at ALL
  - Had a number of threading bugs
- Problems with Java Swing (GUI Toolkit)
  - Documentation relies on implication & example
  - Must be multi-threaded, but not thread safe
  - Failure modes are unpredictable and undocumented
- Eclipse JDT Bugs
  - "[Bug 163680] [1.5] [compiler] JDT Internal Exception under import circularity"
    - Already fixed, should be fully released in February
    - May be masking other bugs
  - Affects compilation of our code
    - Full recompile often required, but this is slow over 600+ classes
- Javadoc crashes
  - Incorrectly extracts class hierarchy from our code
  - Throws a "NullPointerException" and dies
  - Currently unresolved, no workaround yet

13

12/14/2006

RADTools

## Conclusion: Report Card (1)

- ☑ Control & Configuration
  - ☑ Lighttpd
    - ☑ FastCGI dispatchers
    - ☑ stop/start/restart
    - ☑ Almost the entire lighttpd.conf
  - ☑ RoR
    - ☑ Deploy a website (copy files)
    - ☑ Set database & memcached settings
    - ☑ Run dispatchers
  - ☑ MySQL (Start/Stop/Restart/Config)
  - ☑ Memcached (Start/Stop/Restart)
  - ☑ HAProxy
    - ☑ Application pools and configuration
    - ☑ Liveness detection
    - ☑ Start/Stop/Running restart
  - ☑ VM (Start/Stop/Restart)

14

12/14/2006

RADTools

## Conclusion: Report Card (2)

- ☐ Monitoring
  - ☐ Get CPU/Mem/Disk/Network usage
    - ☑ Periodic updates
      - ☑ Time graph/Progress Bar
      - ☑ Record time series data
    - ☑ Filters & DSP Blocks
      - ☑ Turn number into booleans (big latency = failed, small latency = running)
      - ☑ Moving average
  - ☑ Run a load test and graph results
- ☑ GUI
  - ☑ VM Config
  - ☑ Graphing
  - ☑ Service pool configuration
  - ☑ Aggregation of services
- ☑ Basic
  - ☑ Manage logins & authentication
  - ☑ SSH & SCP
  - ☑ Easily script all of the above from Java

15

12/14/2006

RADTools

## Conclusion: Results

- Painful Lessons
  - Standardization is vital
  - Documentation must be explicit
  - Threading is a mixed blessing
- Results
  - We can recreate over a month of labs in mere hours
  - Adding features and services is easy
    - This is necessary for any tool
    - If you can't learn it in one afternoon, it's not usable
  - Simple enough for an undergraduate to use
    - Haven't tested on professors

16

12/14/2006

RADTools

## Conclusion: Future Work

- Javadocs & Website
  - Javadoc bug significantly delayed this
  - May be impossible until a patch is released
- General RADServices
  - Ours are fairly specialized to this class
  - Wish list: VMotion, VM Cloning, Apache, Windows/IS
- A service description language/input
  - We hardcoded the VMs, PMs and Services for this class
  - Would be very easy to replace this with a text file input
- Ultimately: integration with SML
  - We would provide all the support and help we can
  - A chance to do real world tests instead of simulation

17

12/14/2006

RADTools