Emerging Technologies for the Enterprise

March 28-29, 2007 Drexel University, Philadelphia, PA

Ruby in the Enterprise

Erin Mulder
Chariot Solutions
emulder@chariotsolutions.com

Ruby's Sweet Spots

- General purpose scripting for developers and system administrators
- Startups with rapidly changing requirements and competitive feature-driven development
- Rapid development of small to mid-sized greenfield web applications
- Rich web user interfaces

Making Corporate Inroads

- Light e-commerce
- Internal business applications
- Mid-sized customer/partner-facing applications
- Point-to-point integration glue

The Feedback

- “Easy to learn”
- “Very Productive”
- “Fun”
- “Agile”
- “Ooo Shiny..Ajax”
- “Less code”
- “Faster to market”
- “What do you mean, no stored procedure support?”
- “What's a zombie process and how do I kill it?”
- “Where's my code completion???”
- “FastCGI destroyed my marriage”

What do we mean by “Enterprise”

- Large companies, billion-dollar databases
- Large internal IT and operations organizations
- Large portfolio of applications and data stores
- Many integration points, internal and external
- Heterogeneous environment, but with many corporate standards enforced across all applications

What do we need in an Enterprise?

- Data integrity
- Flexible data mapping
- Integration
- Logging
- Monitoring
- Reporting
- Internationalization
- Deployment
- Availability
- Performance
- Scalability
- Maintenance
- Testing
- Longevity

Data Integrity

The Wish List

- Transactions
- Stored Procedures
- Support for high-end RDBMS platforms and features

Data Integrity - Transactions

OKAY

- Ruby DBI and ActiveRecord both support local transactions
- ActiveRecord automatically wraps save and destroy actions in transactions. Larger transactions need to be managed explicitly.
- Difficult to manage transaction isolation levels
- No support for two-phase commit

Data Integrity – Stored Procs

OKAY

- Straightforward to grab connection, call stored procedures and process results manually
- Rails is opinionated **against** stored procedures
- Difficult to bind stored procedure wrapped tables to ActiveRecord objects
- ActiveRecord isn't nearly as helpful for databases built around stored procedures

Data Integrity – RDBMS Support

OKAY

- ActiveRecord supports MySQL, PostgreSQL, SQLite, Oracle, SQLServer and DB2
- ActiveRecord still targets MySQL 4 functionality
- Ruby DBI supports ADO, DB2, Frontbase, mSQL, MySQL, ODBC, Oracle, PostgreSQL, Proxy/Server, SQLite and SQLRelay
- Can use db driver directly for advanced options, but not usually as recent as JDBC drivers

Flexible Data Mapping

The Wish List

- Legacy data models
- Compound or missing primary keys
- Prefixed, suffixed and abbreviated names
- Views
- Stored Procedures

Flexible Data Mapping - Reality

OKAY

- Specify exceptions in ActiveRecord
- Use custom SQL with ActiveRecord or DBI to interact with stored procedures or very complex schemas
- Construct database views to simplify AR-powered reads
- Perhaps rbatis will help?

Integration

The Wish List

- Point-to-point application integration
- Messaging
- Web Services

Integration – Point-to-Point

GOOD

- Easy database integration, flat file manipulation
- Easy to execute system calls
- Easy to wrap C APIs
- No native CORBA support
- Can use RJB or JRuby to take advantage of Java's integration capabilities

Integration – Messaging

OKAY

- Stomp + ActiveMQ for JMS connectivity
- ActiveMessaging (uses Stomp)
- AMQP support emerging
- Some projects integrating with WebSphere MQ
- Wrap C API to integrate with other vendors
- reliable-msg for Ruby-to-Ruby messaging?

Integration – Web Services

GOOD

- Solid support for the creation and consumption of REST and SOAP web services
- SOAP support with ActionWebService in Rails, or directly in Ruby through soap4r
- No support for WS-* standards
- Not a lot of competing implementations to choose from

Logging

The Wish List

- Speed
- Reliability
- File rotation
- Flexible formats and configuration
- Easy consumption by monitoring and correlation services

Logging - Reality

GOOD

- Built-in Logger class is fine for most simple applications, but not very flexible
- Include log4r for much more control over format, log levels, etc.
- Use external log rotation tools
- Format with log4r for consumption by external analysis tools

Monitoring and Management

The Wish List

- Integration with Nagios, Tivoli, OpenView
- Support for standard network management protocols
- Access to fine-grained details about error conditions, object creation, memory usage, garbage collection

Monitoring – Reality

OKAY

- Many existing tools to monitor Apache, database, operating system, network...
- Emerging tools for visibility inside Rails applications (e.g. fiveruns.com)
- Few Ruby plugins for market leading tools
- No JMX equivalent
- Little visibility/control over Garbage Collection

Reporting

The Wish List

- Fast generation and archiving of reports
- Flexible presentation, including text, HTML, charts, PDFs, handheld display
- Polished report-building user interface

Reporting – Reality

POOR

- Ruport for very basic application-level reporting
  - No visual designer
  - Limited feature set (where Jasper was 4 years ago?)
- Gruff and Scruffy for charting (great visuals, but fairly slow)
- Use JRuby, RJB or other integration for direct access to popular Java reporting engines
- Handle reporting outside of Ruby apps

Internationalization

The Wish List

- Support for multi-byte charsets
- Easy support for localized message bundles
- Advanced formatting utilities including capitalization, pluralization, dates, decimals, currencies...

Internationalization – Reality

POOR

- Ruby's String class does not manipulate multi-byte characters properly
- Use alternative string libraries (e.g. ICU4R)
- Use Ruby-GetText to manage and retrieve localized messages
- If your database I18n implementation is better, consider delegating to it (e.g. for toUpper)

Deployment

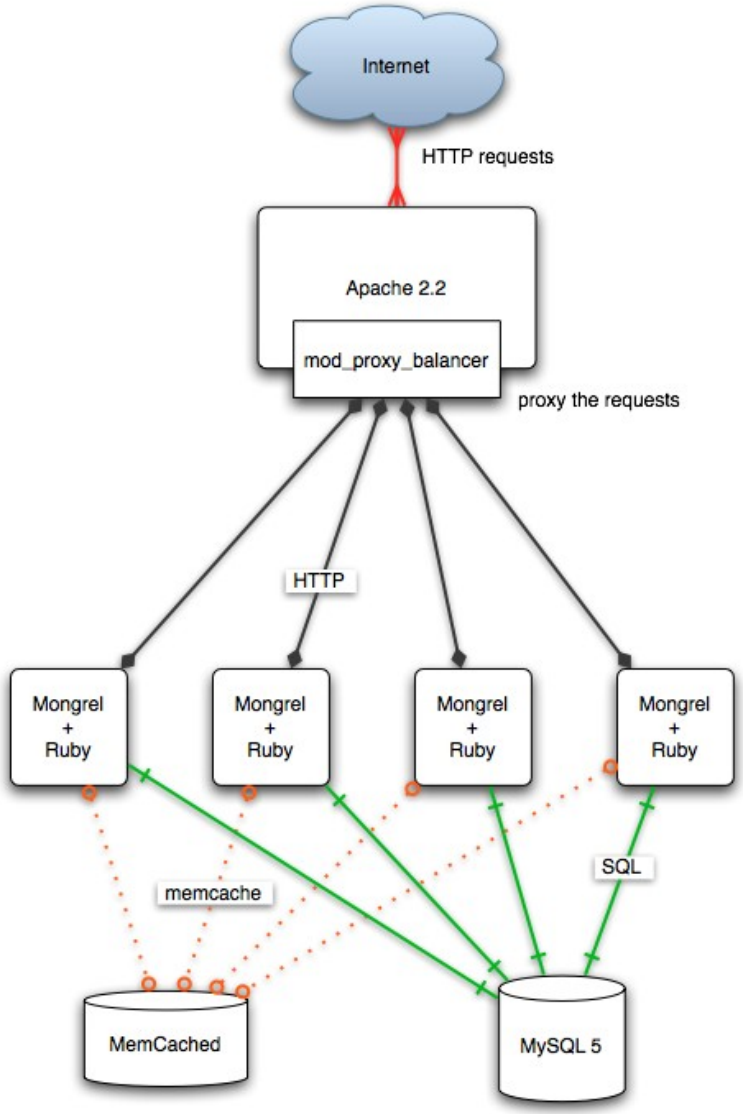
The Wish List

- Multiple, competitive deployment options
- Support for well-known platforms with existing deployment and management utilities
- Utilities for rapid deployment and rollback across large clusters
- Solid package management utilities, with support for multiple versions across apps

Deployment – Options

GOOD

- Current popular favorite is:
  - Apache 2.2
  - mod_proxy_balancer
  - many Mongrel instances
  - memcached
  - shared database
- Many still use FastCGI
- Some use Pound or Pen



Deployment – Platform Support

GOOD

- Some use Lighttpd, Pound, Pen, SQLite, etc. but you don't have to
- Reuse extensive deployment and management tools for:
  - Apache
  - Database
  - Operating System

Deployment – Cluster Mgmt

GOOD

- Capistrano really shines
  - Simplifies mass deployment and rollback
  - Easiest with Unix+Subversion-based projects
- mongrel_cluster for Mongrel clusters
- mod_proxy_balancer easy to configure
- Freeze rails and gems into vendor directory to simplify package management

Availability

The Wish List

- General system architecture maturity and stability
- Fault tolerance
- Cluster and failover capabilities
- Online patch and upgrade capabilities

Availability – Reality

OKAY

- Recommended architecture changes frequently – no long-running case studies
- Apache+mod_proxy_balancer+Mongrel looks fairly stable, but is quite new
- Many key components still under very active development and bug-fix cycles
- Capistrano can help with graceful upgrades

Performance

The Wish List

- General platform and framework performance
- Threading, I/O efficiency, memory efficiency
- Resource pooling
- Database interface efficiency
- Fine-grained concurrency utilities
- Good profiling tools

Performance – Reality

OKAY

- Ruby is slow (though YARV will help)
- Multi-process model is often the only option
- Default Rails does not optimize database usage
- Need to pay careful attention to application code to prevent or fix performance problems
- Offload intensive work to database

Scalability

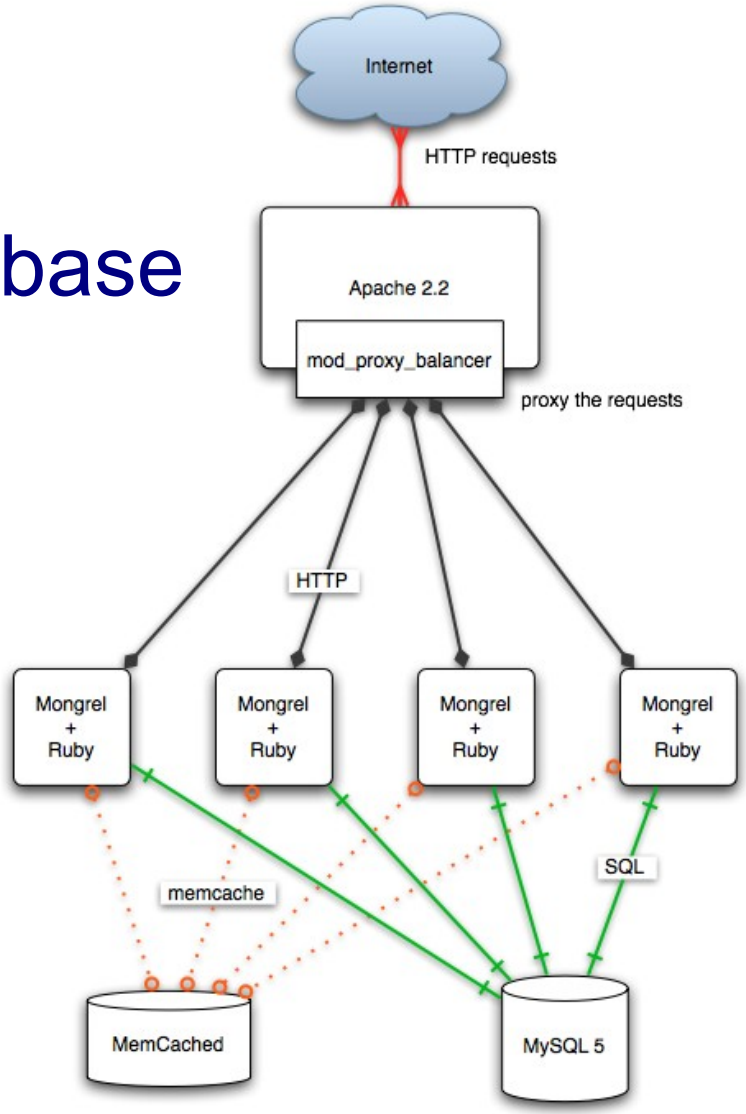
The Wish List

- Near-linear horizontal scaling as additional clone servers are added
- Optimal utilization of the processor, memory and I/O capacity of each server
- No loss of performance or conversational state when participating in load-balanced cluster

Scalability – Reality

GOOD

- Scales out fairly well
- Eventually need to scale database
- Very easy to add new nodes
- Capistrano is great



Maintenance

The Wish List

- Size of codebase
- Velocity of change in frameworks, backwards compatibility policies
- Availability of developers with the right skills
- Learning curve for new developers
- Support contracts

Maintenance – Reality

OKAY

- Generally less code than PHP or Java
- Many libraries are still immature, changing frequently with little backwards compatibility
- Limited supply of skilled Ruby developers and supporting vendors
- Testing culture and readability help new team members, lack of tool support hurts

Testing

The Wish List

- Unit testing and code coverage tools
- Mock utilities
- Ease and resilience of automated functional testing

Testing – Reality

GOOD

- Excellent support for unit testing
- rcov analyzes code coverage
- FlexMock makes mocking easy
- Many automated test tools
- Rails URL patterns make it easy to use any external test utility (GUI, non-Ruby, etc.)

Longevity

The Wish List

- Large following and installed userbase
- Diverse group of vendors offering implementations, tools and support
- Strong, business-friendly leadership

Longevity – Reality

OKAY

- Ruby and Rails are primarily driven by small groups of open source developers
- Ruby leader is on no time table. Rails leader is opinionated, biased against enterprise features
- No large corporate sponsors funding resolution of enterprise concerns
- Some attention from Sun and Microsoft for supporting Ruby in JVM and CLR

So Is Ruby Ready?

- **YES** – small to medium-sized, user-facing business applications
- **YES** – glue to integrate or service-enable legacy applications
- **YES** – public-facing web applications that are not mission critical or insanely high volume
- **YES** – aggressive time-to-market deadlines
- **YES** – utility services on message bus

So Is Ruby Ready?

- **MAYBE** – small to medium, mission-critical applications (with extensive testing)
- **MAYBE** – integration with complex data models
- **NO** – performance-critical infrastructure
- **NO** – large, mission-critical applications
- **NO** – critical internationalization requirements
- **NO** – apps requiring distributed transactions

Looking ahead

- Have we seen this trend before? (Java, 1997)
  - Slow, with future VM improvements offering hope
  - Gaining traction as a UI and web technology, but not yet considered enterprise-worthy
  - Front of the pack in latest Internet paradigm switch
  - Missing libs/features that already exist in other languages
- Or maybe not?
  - No big commercial backer
  - No commitment to enterprise features

Looking ahead

- A new landscape?
  - Sun hired main JRuby developers
  - Microsoft hired main RubyCLR developer
  - The main Ruby implementation is dropping support for continuations and green threads, two of JRuby's biggest challenges
  - Sun knows how to create fast VMs
  - What happens when Ruby has fast, competing implementations from both Sun and Microsoft?

Mentioned along the way...

- Ruby DBI
- rbatis
- RJB
- JRuby
- Stomp
- ActiveMQ
- ActiveMessaging
- reliable-msg
- soap4r
- log4r
- FiveRuns
- Ruport
- Gruff
- Scruffy
- ICU4R
- Ruby-GetText
- mod_proxy_balancer
- Mongrel
- memcached
- Capistrano
- mongrel_cluster
- FastCGI
- Pound
- Pen
- YARV
- rcov
- FlexMock
- RubyCLR
- Pound
- Pen