Scaling Up Without Rewriting Your Core

By
Eran Sandler
@erans
http://eran.sandler.co.il

Lecture Idea by Yaniv Golan

IL Tech Talks Week
June 30th, 2011

We will talk about...

- Your Options When The Site Slows Down
- Consequences of Full Rewrites
- Things to consider before rewrites
- What we did at AOL Answers

But first,

Some hard facts...

Scaling Is HARD!

- It takes time
- There are no magic solutions
- A lot of rinse & repeat
- Ongoing, never-ending task
- One size doesn't fit all

Let's Start...

So...Your Site Slows Down...

- If it's not a bug - Smile :) You are growing!
- But what can we do?

- We can:
  - Throw more servers at the problem
  - Optimize the code
  - Rewrite some (or all) of the code

Rewrites Can Be Dangerous!

- Can take longer than planned
- Can introduce new bugs
- Proven and debugged code is thrown away
- Special logics can get forgotten in the process

It's always good to learn from others!

Consequences of Full Rewrites
GIS System at the Army
Background

- Geographical Information System (GIS)
- Native GUI Based 2-tier Windows Application
- Monolithic Delphi code
- 1000s of install seats, 100s of install sites
- No connection between sites. No Internet. Nothing.
- Mostly hard coded for the original tasks

Consequences of Full Rewrites
GIS System at the Army
New Requirements

- More robust and extendable code (new requirements are coming every day)
- Less rigid data access layer (it took a long time to add new types of objects)
- Multiple DB support (Paradox + Oracle)

Consequences of Full Rewrites
GIS System at the Army
Results

- Estimated at 3 months - took about one year
- No new features or releases of old version during the time of rewrite
- The unit had a rough time explaining this long delay

Consequences of Full Rewrites SmarTeam Next Gen (STiNG) Background

- Product Life Cycle (PLM) management software
- 2-tier Native Windows Application
- Mostly Monolithic Delphi Code
- Dynamic Data Model (can add object types without code releases)
- 100,000s of seats, 10,000s install sites

Consequences of Full Rewrites
SmarTeam Next Gen (STiNG)
New Requirements

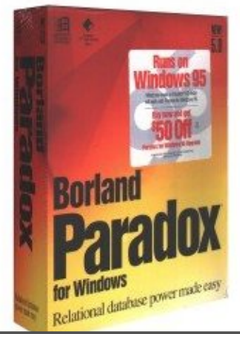
- Avoid Installation at Client Side (as much as possible)
- Support thousands of seats from a single server
- Cross-Platform Support (for integration with non-Windows CAD & ERP systems)

Consequences of Full Rewrites
SmarTeam Next Gen (STiNG) Results

- No major features were introduced to old code for almost a year. **The company almost closed** due to that.
- New code originally written in Java
- Java Applets eventually became server side visual components
- Eventually, Java code abandoned and server side code re-written in the (then) new .NET Framework with COM and P/Invoke wrappers around older code
- Client code re-written in C/C++
- Client/Server communication developer pre-SOAP and had to be adapted

Famous Rewrite Failures

- Netscape - rewrite for Netscape 4.x
- Digg - complete rewrite for Digg 4.0
- Borland Paradox for Windows - complete rewrite for Windows 3.0 + new GUI and OO language



What We Did at AOL Answers Background

- AOL Answsers (http://aolanswers.com) is a Q&A web site that proactively tries to help you find answers to your questions
- Uses a self developed semantic engine to understand the question and find possible matching users that can answer the question
- August 2006 Launched originally as Yedda (People · Sharing · Knowledge)
- November 2007 Acquired by AOL

Aol Answers.

What We Did at AOL Answers
Iteration 1
Original Code

- Microsoft Stack (Windows Server, ASP.NET, SQL Server)
- Written pre-ASP.NET MVC and pre-AJAX.NET

- We took the cautious approach (pre optimization is the root of all evil!)
- Limited per request caching - until code and database settled a bit and we figured the access patterns
- Hardware Configuration: Hardware load balancer, 3 web servers and 2 DB servers (active/passive replication)

Aol Answers.

What We Did at AOL Answers
Iteration II
Memcaching

- Traffic, users and load increases
- Code is more stable, DB access patterns stabilized
- Memcached servers added
- Relevant load hooks were added and a framework to automagically figure out which cache keys to invalidate
- Hardware configuration changes: added 2 memcached servers

Aol Answers.

What We Did at AOL Answers
Iteration III
Proxy Caching

- Post AOL acquisition - massive integrations throughout the AOL content sites network using Javascript widgets and RSS feeds
- Widgets required to load as fast as possible
- Problems in our servers should never affect AOL sites

Aol Answers.

What We Did at AOL Answers
Iteration III - Continued

- Added Varnish - an HTTP accelerator (http://varnish-cache.org)
- Started serving Widgets and RSS feeds via CDN
- Hardware Configuration Changes: added 2 Varnish servers (active/passive configuration using heartbeat) in a different data center
- Handled 250M monthly widget requests (~100 reqs/sec) with only 5 web servers

Aol Answers.

What We Did at AOL Answers
Iteration IV
Apache SOLR

- Semantic engine's custom built clustering algorithm was struggling to scale with questions and users
- Replaced original clustering algorithm with Apache SOLR and custom queries
- Hardware Configuration Changes: added 3 SOLR servers (write master, 2 read slaves)

Aol Answers.

What We Did at AOL Answers
Iteration V
Offloading Computation

- Some tasks during live user requests were not required to be performed during the live request itself
- Offloaded the tasks using a queue service (RabbitMQ) to worker processes on non web server machines
- Hardware Configuration Changes: 2 Async servers running queue service and worker processes

Aol Answers.

What We Did at AOL Answers Iteration VI

- Load balancer needed replacement to handle TCP connections load
- Original design created too many memcache server requests per HTTP request (100s of memcached requests)
- Dropped Hardware LB in favor of Varnish (used it as both caching server and load balancer
- Added new backend server to use different DB fetching and caching strategy to reduce memcache access to a minimum (4-5 memcached reqs/HTTP request)
- First "major" code rewrite in ~4 year of original production code

Aol Answers.

Take Aways

- Full rewrites can (and probably will) take more time than you expect
- Consequences can be harsh - up to a point where a company can close its doors
- Full rewrites can be avoided in most cases and there are lots of options to get more bang per $$$ with little to no code changes
- Understanding your code and its usage patterns are essentials for fixing problems and avoiding rewrites
- Measure, measure, measure!

Thank You!

References

- Varnish - HTTP Accelerator - http://varnish-cache.org
- Apache SOLR - http://lucene.apache.org/solr
- RabbitMQ - http://www.rabbitmq.com
- Kestrel - https://github.com/robey/kestrel

Appendix
A Few Technical Conclusions

- Event based HTTP servers are MUCH better for non computational intensive requests. Gives more Bang/$$$
- Offload any non critical computation to worker processes on different machines
- Caching speed up things but managing it is hard work!
-