Igniting the Spark: Building Online Services for Borderlands 2

Jimmy Sieben
Lead Programmer, Gearbox Software
A Word About Me

- I’ve been programming for 20+ years, 15 professionally
- Making games since 1995; at Gearbox for over 10 years
- Network Programming on multiple titles
  - Halo: Combat Evolved (2003: PC)
  - Brothers in Arms: Road to Hill 30 (2005: PC/Xbox)
  - Borderlands (2009: PC/PS3/Xbox 360)
  - Borderlands 2 (2012: PC/PS3/Xbox 360)
A Word about Borderlands

• Introduced in 2009, sold over 6 million
• Coop Shooter Looter:
  • FPS Action, Action-RPG Mechanics
  • 4 player Cooperative, drop-in drop-out
• Borderlands 2 released in 2012, sold over 6 million
  • Refined Shooter Looter, enhanced coop play
  • Built SHiFT and Spark to connect to community
  • A new initiative, something we’ve never done before
What is Spark?

• **Spark**: Our backend platform
  • Internal name

• **SHiFT**: Our online service
  • Customer-facing
Spark Features for Borderlands 2

- **Archway**
  - SHiFT Account Signup, Platform account linking / authentication (Xbox LIVE, PSN, Steam)
  - Code Redemption & Rewards
- **Discovery**
  - Dynamic configuration, per-environment and per-user
  - Supports user populations for betas and testing
- **Micropatch**
  - Data-driven hotfixes for rich game content
  - Hard-coded or service-delivered (tied into Discovery)
- **Leviathan**
  - Telemetry for gameplay
  - Stats and Events with rich metadata
Why do this?

• Games are increasingly social, connected experiences
• AAA Games must go beyond the box
  • Embrace the web and mobile, companion experiences
  • Engage with players any time, anywhere
  • Build the brand
• Ultimately, all about the customer
  • Build the connection directly to the fans
  • Enable the community to forge connections
• These are pillars of the next generation of games
Archway: Accounts and Authentication

• Single Sign On via ticket verification
  • Xbox LIVE, PSN, and Steam supported
• Platform ID only comes from a valid ticket
  • This makes it difficult to impersonate a user
Spark: Single Sign-On Process

Game: Acquire User Ticket from Platform API

Game: Send ticket to Spark

Spark Backend: Verify ticket
Spark: Reward Redemption

• Built a system of Offers and Entitlements into our account system
• Created a code generator for 5x5 codes
SHiFT Code Example

Xbox 360  CBKBJ-3TXBH-55S3X-W6TT3-9BSZ5

PlayStation 3  WBKTB-CB6TT-X6WCJ-9T5BB-W5CBT

Steam  WTCTB-HHX3C-39FJJ-JB333-FWWJZ

These are real-live codes, good for a Golden Key in Borderlands 2. Redeem them when you get home 😊
SHiFT Code Reward Architecture

Offer

Offer Text

Entitlements

Entitlement 1
GoldenKey
Consumable: 1

Entitlement 2
ShiftCustomization
Valentine’s Skin

Entitlement 3..n
Entitlement Name
Parameters
SHiFT Code Entitlement Architecture
SHiFT Code Redemption Patterns

1 Shift Code

- Single Redemption
- Fixed Redemption (small)
- Fixed Redemption (large)
- Timed Redemption
- ???
Spark: Micropatching

- Borderlands 2 is a rich data-driven game
- So much of the game is actually implemented in data, how can that be updated on the fly?
- We built a system to package data updates into Micropatches, deliver them via Spark
- This allows us to do balance tweaks, bug fixes, live events by changing data implementation
- Authoring support in editor and backend tools
Spark: Telemetry

• How do players experience Borderlands 2?
• Drive Micropatches
• Provide business intelligence during launch
• Get visibility on exploits and cheats
• Feed into future development
How do you do this?

• What does a backend service look like?
  • GDC talks
  • HighScalability.com
  • Amazon, Facebook, Microsoft, Google publications
  • Phone a Friend

• How do you choose technology for Blue Sky?
  • Know your priorities: What you like, Healthy ecosystem
  • Rapidly evolving space
  • Just choose and go – adaptability is key
The Challenge of AAA

Startups & mobile teams reference soft launch, gradual run-up to inflection point

(John Mayer tweets about Words with Friends)
The Challenge of AAA

• AAA game launches are the opposite:
  Vertical, long tail and plateau
Building the Service

• Research
• Build a team
• Start coding
• Ship it 2-3 years later?
• .... This isn’t easy. Is there a better way?
Building a Beta!

• We shipped BTest for Borderlands on Steam
• Beta test of our toughest feature: Telemetry
• Early visibility into key decisions and questions for Authentication
• The focus of the first 10 months of Spark
• Shipped on 9/9/2011
BTest Postmortem: Test Everywhere

• We took our beta to different networks
  • 2k and Gearbox corporate
  • Home, with and without VPN
  • QuakeCon!

• QuakeCon used a transparent Squid proxy
  • Exposed a copy/paste bug in our HTTP code
  • Oops, sending POST data on a GET!
BTest Postmortem: Crash Bug!

• Clock synchronization problem on server
  • Game clients slowly drifted away from server
  • Some crash reports early
  • By Saturday morning, all clients crashing
  • Workaround server side, instantly fixed crashes!

• Lessons
  • Some test are vectors very difficult to predict
  • Server tunability is incredibly valuable
  • Tuesdays are the Best Days! (Not Friday!)
BTest Postmortem: Leaderboard

• We created a simple leaderboard to encourage players to try the update
  • It got slower and slower and slower, until updates were taking over 45 minutes. Refactored queries and got updates down to 20 seconds
  • Hacker submitted bogus data within hours of launch

• Lessons:
  • Test with full data set early
  • Try and break your assumptions
  • Malice is the Norm
BTest Postmortem: Database Schema

• We didn’t know exactly what we wanted to ask, so we built a very flexible, generic model
  • Very quickly got too slow to work with
  • How many enemies killed: 1 hr+ query times!
  • Database size out of control
  • Hard to plug in tools for visualization

• Lessons:
  • Knowing what you want to do w/ data is crucial
  • Data archival was very useful
BTest Postmortem: Capacity Planning

• Looked at Steam data in March
• Predictable decline to July Launch
BTest Postmortem: Capacity Planning

- We shipped Btest in September...
- Steam Summer Sale!
- Borderlands 2 announced!
BTest Postmortem Capacity Planning

• Scrambled to handle dramatically higher load
  • Resized DBs, more servers, reconfiguration

• Lessons:
  • Pay close attention and adjust constantly
  • Be plugged in to PR and Business
  • Be agile
Building another Beta!

• BTest was so helpful, we shipped another beta!
• Launched December 13, 2011 (a Tuesday!)
Gearbox Moves into the Cloud

- BTest1 was hard to operate on shared hosting
  - Capacity hard to adjust, and we didn’t get it right
  - We knew we needed to design for more flexibility
- BTest2 Shipped on Amazon Web Services
  - EC2, RDS, ELB
  - Steep learning curve, but paid off...
  - Didn’t get everything right...
BTest2 Postmortem: Holiday Stability

- We launched and were mostly stable
- However, problem Christmas evening!
  - Our game was still selling, new people playing
  - Queues were backing up, not severe
  - A few days later, CPU is pegged!
  - The Cloud to the rescue! Deploy more bigger!

- Lessons:
  - Queue storage in cloud gave wiggle room
  - It was actually pretty easy to recover from CPU peg
  - Capacity planning still hard!
BTest2 Postmortem: Missed Opportunities

- New to AWS, Deployed regular EC2 instances
- Skipped VPC
  - This turned out to be a mistake
  - More difficult to secure some resources like we wanted
  - Had to build load balancing logic into app layer
- Lessons:
  - Embrace as much of the feature set as you can
  - Don’t be afraid to choose long term over short term
  - Especially for a Beta!
Going Wide

• After shipping two iterations, had some confidence in architecture
• Define the final feature set for the game
• Building an implementation plan is hard
  • Include all stakeholders
  • Navigate difficult policy waters
  • Finish building the team and finish the code!
Discovery: Design for Tunability

- Created **Discovery** service which is central store for service configuration
- If the client doesn’t get service info, disable it
- Flexibility is key:
  - Simple key/value pairs, dirt-simple format
  - Different configs for environments, titles, and platforms
Discovery: Design for Testability

• Endpoint URLs and Versions from Server
  • Single point of entry hardcoded in code / platform config
• Client overrides for configuration via INI and command line
  • Allowed developers to work offline
  • Allowed QA to customize things for testing
  • Supported even in final builds
  • Enabled internal and external beta scenarios!
Design for Bad User Behaviors

• Be conscious of user behaviors which can harm the service
  • Signup / Signin
  • Code Redemption
• We implemented client side throttles
  • Prevent button mashing denial of service attacks
• Don’t create incentives for users to be bad
  • Signup/delete path not optimized for churn
  • Signup rewards once per platform ID, ever
Load Testing Difficulties

• We didn’t fully understand how ELB worked at first
  • Amazon’s documents are good, but easy to overlook
  • Turn off DNS caching
  • Prewarming critical
• JMeter limitations led us to write custom scripts
• Costs can get out of hand! Watch carefully!
  • Should have invested more in automation
Launching Borderlands 2

• Borderlands 2 launch: September 18, 2012
• The team was all set in the war room
• Night and day shift established for launch
• Latest capacity info from industry friends and experts projected we would survive
  • But still, wave of terror washed over me a T-6 hrs
So?, What Happened

• Nothing!
• Well, almost. At least at first.
  • Email queues backed up, deployed more workers
  • Slowing turning up the dials on telemetry
  • Watching the numbers, generally OK
  • Went to sleep happy!
• A few things did come up in the following days...
Day 1: Bad Query

Beware tolower() query on indexed columns
Our testing missed this because it only shows on a loaded database
Day 2: Keeping Telemetry Going

• Launch week capacity was tough to manage
• We wanted to keep costs in check, but had not implemented AWS Auto-Scaling Groups
• Manually add/remove instances at set times
Day 2: Keeping Telemetry Going
Day 2: Operation Mistake Reveals Bug

• Made a mistake pushing change to Redis
  • Lots of users kicked off service accidentally
  • Many of them didn’t come back automatically!
Day 2: Telemetry Bug
Day 2: Telemetry Bug

• Actually found 5 different bugs  
  • Reauthentication is a tricky process to test  
  • Bugs in each layer involved in process  
  • Hard to test all the interactions in the system  
• Fixing it was also a challenge  
  • Coordinate operations, dev, production, QA  
  • More on that later...
Day 3: Bad Build

- An early Steam patch was released with bad configuration, pointing at test environment!
- Fortunately, just after patch release we took down test environment for maintenance.
- Deployed a workaround on server frontdoor
- Could not bring it back up until all users were patched
Day 3: Bad Build Continues!

- It wasn’t just our main SKU that had this problem
- We released that code to the Russian SKU as well, which had a delayed update plan
- It took almost 3 weeks to deploy that patch
- **Still** not out of the woods.
  - Unpatched users still out there
  - Pirated copies? Users that didn’t update?
- Lesson: be very careful with configuration!
SHiFT Codes!

• A few days into launch, we were stable enough to start using our SHiFT Codes
  • Randy Pitchford (@DuvalMagic) got things started with some quick tests
  • Engaged directly with devops team to measure results
  • Got a little TOO engaged...
SHiFT Codes: Chaos
SHiFT Codes: Chaos

• While looking for real-time code redemption results, I issued a bad query that impacted some monitoring.
• Took most of the afternoon and evening to recover:
  • Redis failover scripts did not work as expected
  • Restart monitoring node, stabilize cluster
  • Move some monitoring functionality to new node
• Lessons:
  • Try not to intermingle monitoring for different components
  • Be extra careful querying 100MM record datasets!
SHiFT Codes: Optimizations

• During week 2, we used a new type of code that was active for a period of time
• This allowed us to build a social engagement strategy around code redemption times
• Essentially created Flash Mobs for SHiFT
SHiFT Codes: Optimizations

Average response time, broken down by tier (ms)

Average: 21 ms
SHiFT Codes: Optimizations

Average response time, broken down by tier (ms)

- GC Execution
- Ruby
- Database

Average: 52 ms
SHiFT Codes: Optimizations
SHiFT Codes: Optimizations
SHiFT Codes: Optimizations

- Linear table growth as redemptions increase
  - By the second weekend, there were many many rows
  - Lookups on this table were missing an index
  - Didn’t catch this growth over time in testing
- Query for entitlement and offer info not cached
  - This data doesn’t change during a code drop
  - Missed this: requires large table and flash mob
  - Easy to cache in app layer to reduce DB pressure
SHiFT Codes: Unexpected Behavior

Telemetry traffic pattern changes when a code drops
Users Save & Exit game, wait to redeem in menu
Causes spike and lull in telemetry traffic
SHiFT Codes: Unexpected Behavior

- We didn’t expect this behavior
  - Fortunately this didn’t cause a big problem
  - Had extra capacity on hand for launch window
- Lessons
  - Carefully think through interactions of systems
  - However, must recognize that not all user behaviors can be predicted
  - Be prepared for surprises
The Challenge of AAA: DLC

• Successful AAA games must have a DLC plan
• Borderlands 2 successfully launched 5 packs
  • Week 4: Gaige: Mechromancer character class
  • Week 5: Captain Scarlett & Her Pirate’s Booty
  • Week 10: Mr Torgue’s Campaign of Carnage
  • Week 17: Sir Hammerlock’s Big Game Hunt
  • Week 18: Domination, Madness, and Supremacy
The Challenge of AAA: DLC

• This put a lot of pressure on live team
  • New Features, bug fixes, balance tweaks
  • Packaging and builds
• Meanwhile Spark team had challenges too
  • Improve service manageability, security patches, keep it running
  • Optimize server performance and costs
• Lesson: Very difficult to do things post-launch
  • Plan ahead for anything you need in the first 10 weeks
  • Lots of slack in schedule, communicate with dev
What Didn’t Happen?
What Didn’t Happen?

• Spark performed above expectations
  • No downtime in launch window
  • User-facing components didn’t buckle
  • Team wasn’t killed keeping it going
Team Wasn’t Killed Keeping it Going
Why Did We Succeed?

• Great team of smart, passionate, and committed people
• Spent adequate time testing user-facing (10 weeks from certification to launch)
• The cloud: over provision, then scale back
• We launched 2 betas
What’s Next?

• We originally developed Spark for a single title, Borderlands 2
  • Felt that constraining scope was key to success

• SHiFT was so successful, we shipped it again!
  • Aliens: Colonial Marines launched February 2013
  • Integrated accounts, rewards, and built new rewards for SHiFT
Getting Spark to 1.0

• Finish integrating Amazon features: VPC & ASG
• Optimize costs around known traffic patterns
• Enhanced monitoring and deployment tools
• ... And beyond!
  • Find new ways to use the platform!
  • Experiment with new ways to engage with fans
Final Thoughts

• This is really important for next-gen
• Capacity planning is very hard to do
  • Coped by being agile and relying on the cloud
• Ship early and often
  • Possible even for AAA Games!

Jimmy Sieben @jimmys
jimmy.sieben@gearboxsoftware.com