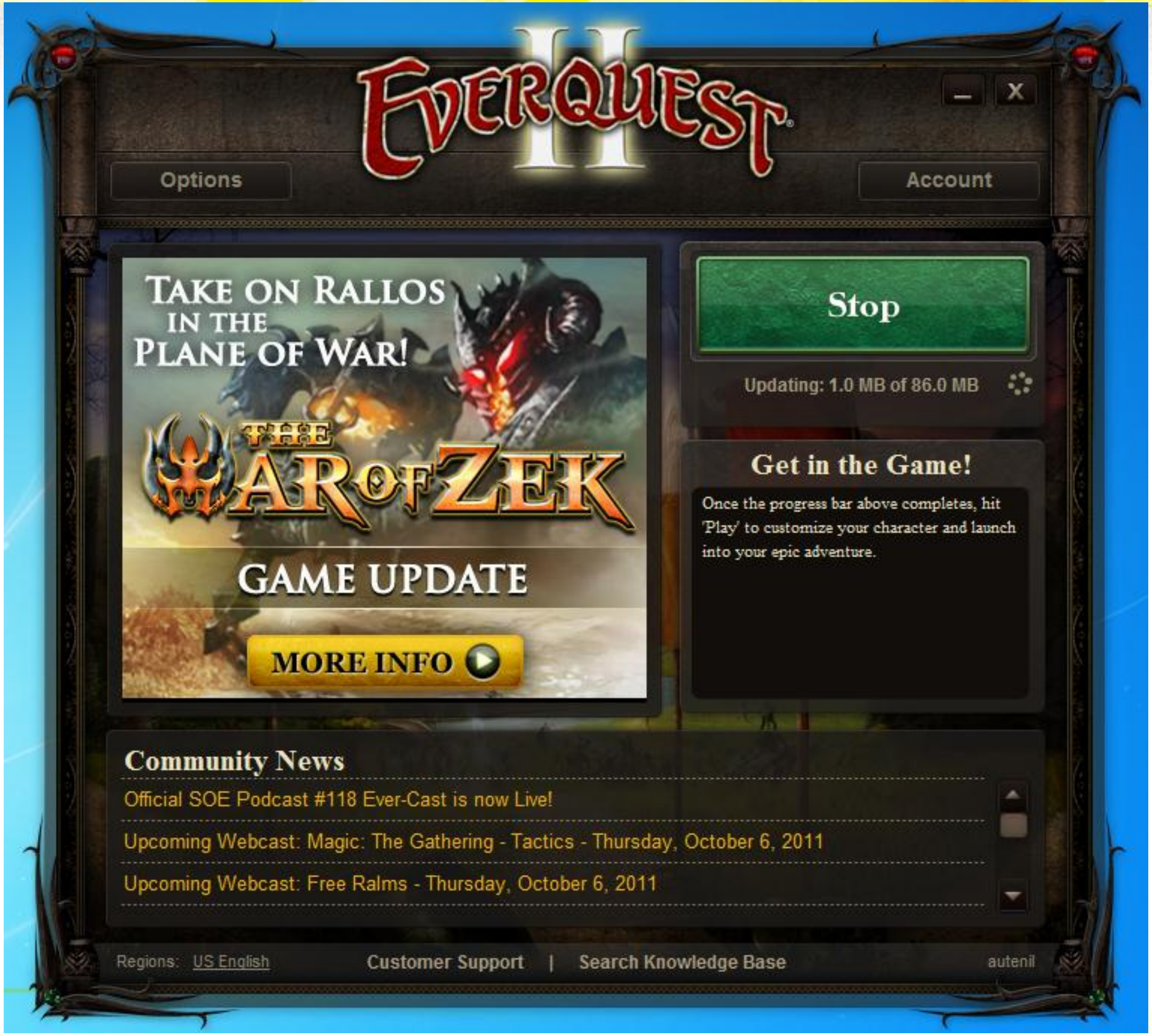
EverQuest II Extended: Streaming a Non-Streaming Game

Joshua M. Kriegshauser

Technical Director, Sony Online Entertainment

jkriegshauser@soe.sony.com
@autenil
http://kriegshauser.blogspot.com/

Game Developers Conference® Online 2011
October 10-13, 2011 | Austin, TX
www.GDCOnline.com



Updating: 1.0 MB of 86.0 MB

Get in the Game!

Once the progress bar above completes, hit 'Play' to customize your character and launch into your epic adventure.

Community News

- Official SOE Podcast #118 Ever-Cast is now Live!
- Upcoming Webcast: Magic: The Gathering - Tactics - Thursday, October 6, 2011
- Upcoming Webcast: Free Ralms - Thursday, October 6, 2011

EverQuest II (Sep 28 2011 10:35:48) USER OPTIMIZED: SOEBuild=8071L

Caching appearances

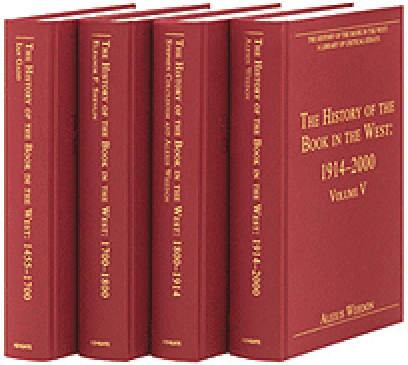
65%

2.3 MB

== ?

Some History

- Full Download: 12.5GB
- Download Time:
  - – 1h 45m (2MB/s)
  - – 7h (512KB/s)
- Poor Free Trial stats

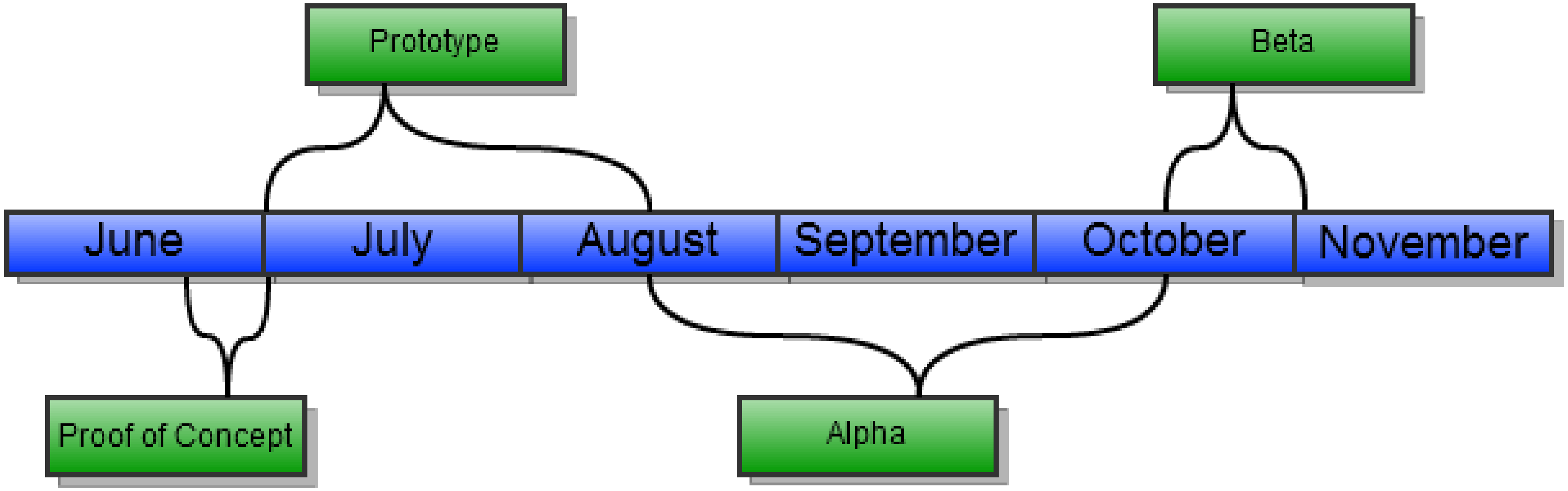


- Wouldn’t it be nice if…

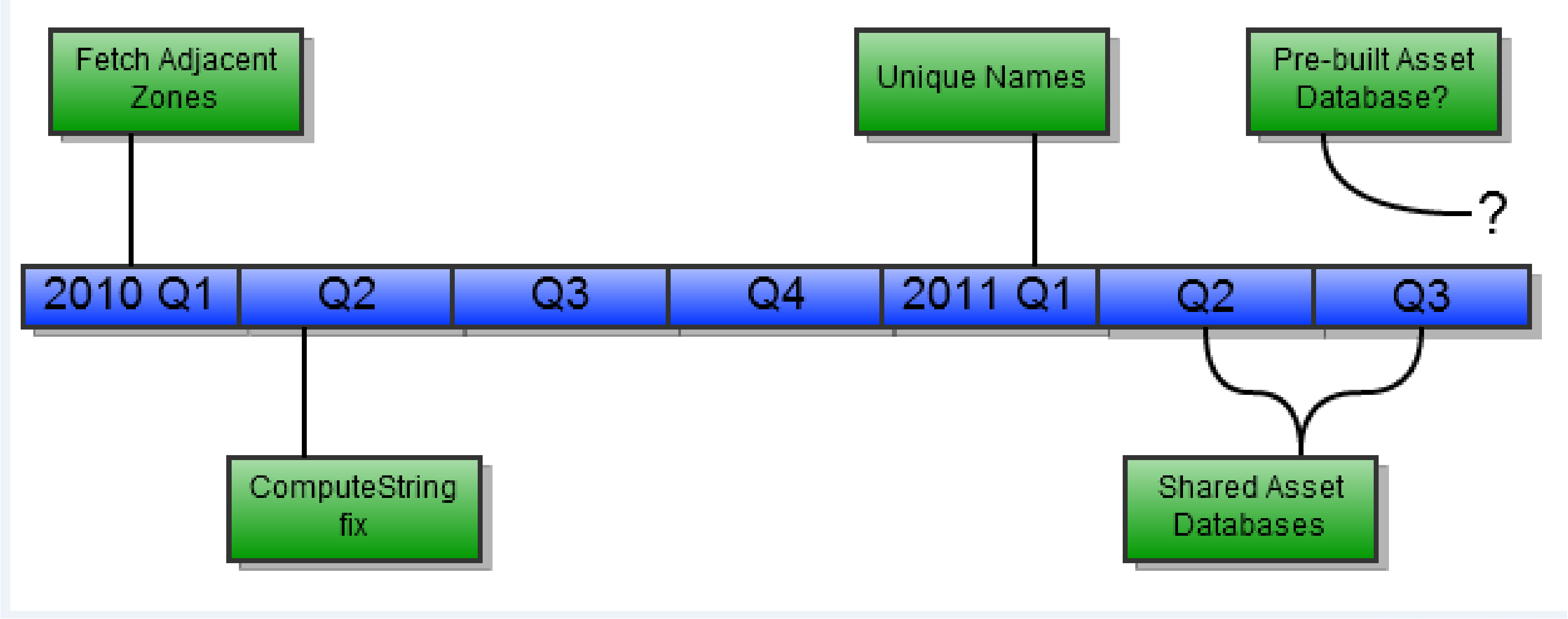
Initial Goals

- Entire game playable
- Initial download < 200MB
- Playing within 20 minutes on 512KB/s

Timeline: 2009



Timeline: 2010-2011

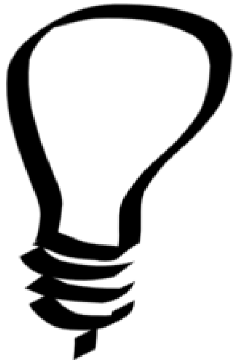


Hurdles

- Serving the assets
- Getting the assets
- Storing the assets
- Maintaining frame-rate

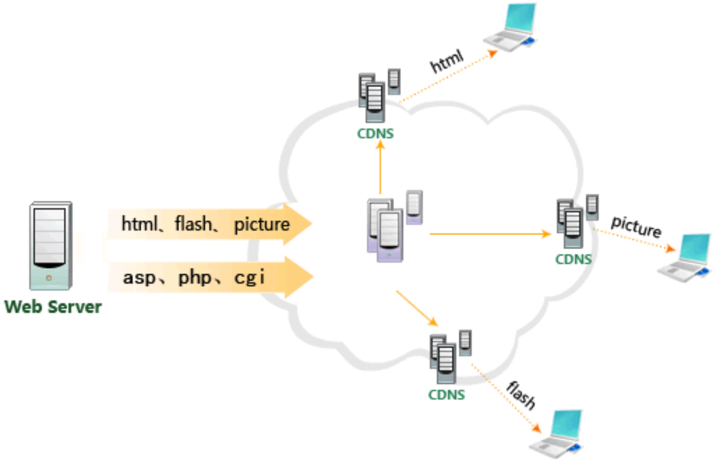
Serving the Assets

- EQII has 507,295 client-side assets
- Initial thought: Custom Server
  - Always serves the latest assets
  - Understand data relationships
  - Binary data over UDP
- Cons:
  - Must build own infrastructure



Serving the Assets

- Alternative: Content Delivery Network (CDN)
  - – Infrastructure already exists
  - – Caching/availability
- Cons:
  - – Involves third party
  - – HTTP (text) over TCP
  - – Certain … quirks
  - – No game data knowledge



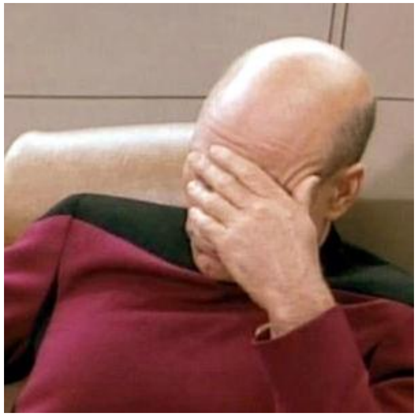
Serving the Assets

- Asset Manifest
  - Unique Identifier, size and version per file
  - Binary vs. Text
- Unique ID: 64-bit filename hash
- File Size: 16-bit (file size in KB)
- Version: 16-bit file data CRC
- 507,295 x 96 bits = ~6MB full manifest

LFMF: Default Arguments

```
CRC16::ComputeMem(data, len, startcrc=0);
CRC16::ComputeString(data, startcrc=0);
```

Don’t use ComputeString()
when you mean ComputeMem()

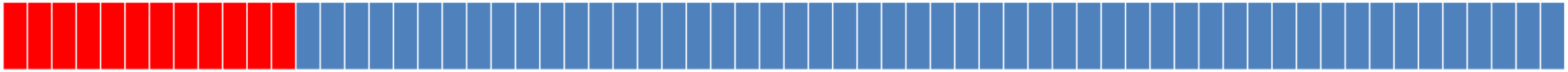


Serving the Assets: Manifests

- Initial plan: Manifest patched via Streaming
- Initially based on custom server method
  - Client uploaded CRCs of 4096 manifest sections
  - Server responded with updated manifest patches
- Manifest.crc always downloaded (~8KB)
  - 4096 x 16-bit manifest section CRCs
  - Sections patched when CRC didn't match
- 6MB Full Manifest -> 200-300KB patch

Serving the Assets: Manifests

- Manifests sorted by filename hash (64-bit)
- Manifest divided into 4096 (2^{12}) sections
- Top 12 bits indicate section



- Each section had its own sub-manifest

  0000/manifest – 0fff/manifest

- Client downloaded necessary sub-manifests

Serving the Assets: Manifests

In-Client Manifest Patching	
PROs	CONs
Manifest always reflects current assets	Code Heavy
	CDN Peculiarities

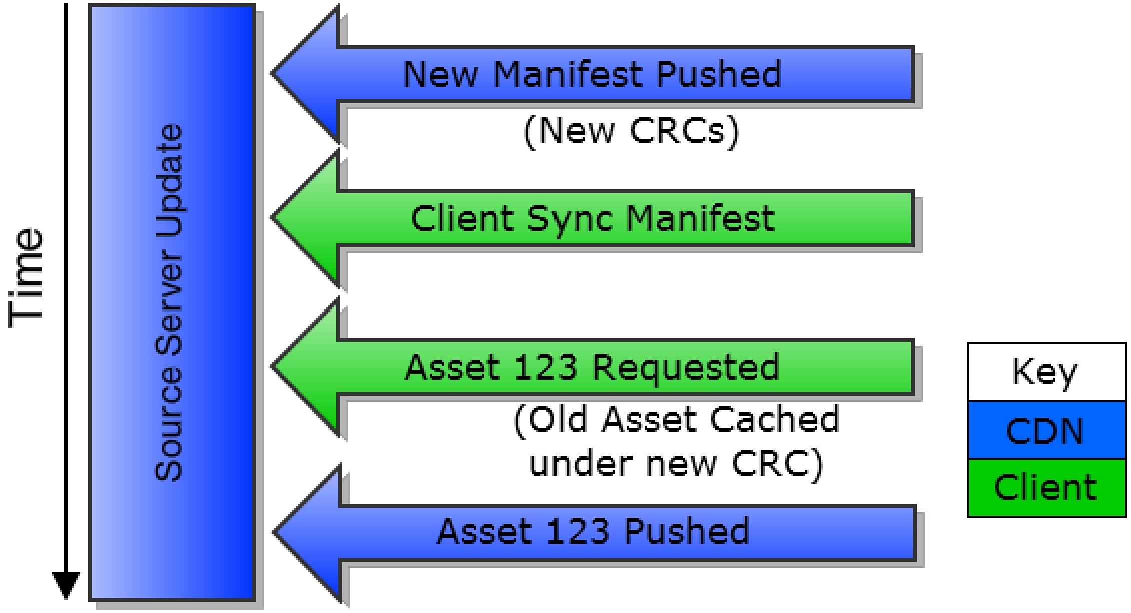
- Current system: Controlled by Patcher
  - Can be delta-patched
  - No additional code support

CDN Peculiarities

- Source Server
- Caching nature good… usually.
- Cache Timeout
- URL Decoration: <unique ID>?CRC=12ab
  - Cache based on unique URL
  - Doesn’t guarantee uniqueness

CDN Peculiarities

- Race Condition



- Fix: Unique Files including CRC: manifest-12ab

Uploading Assets

- Assets divided into 4096 “buckets”
  - Same as manifest sections
- Filenames on the CDN: 64-bit name hash

Asset: ui/default/images/icons/icon_is64.dds

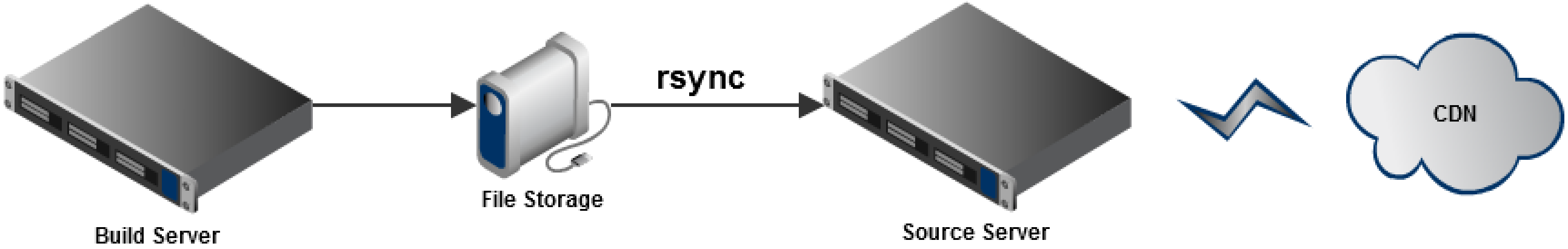
64-bit CRC: 3c9431aab0133431

CDN file: 03c9/3c9431aab0133431-12ab

- Full HTTP request path:

http://eq2.patch.station.sony.com/patch/eq2/assets/common/03c9/3c9431aab0133431-12ab

Uploading Assets



Getting the Assets

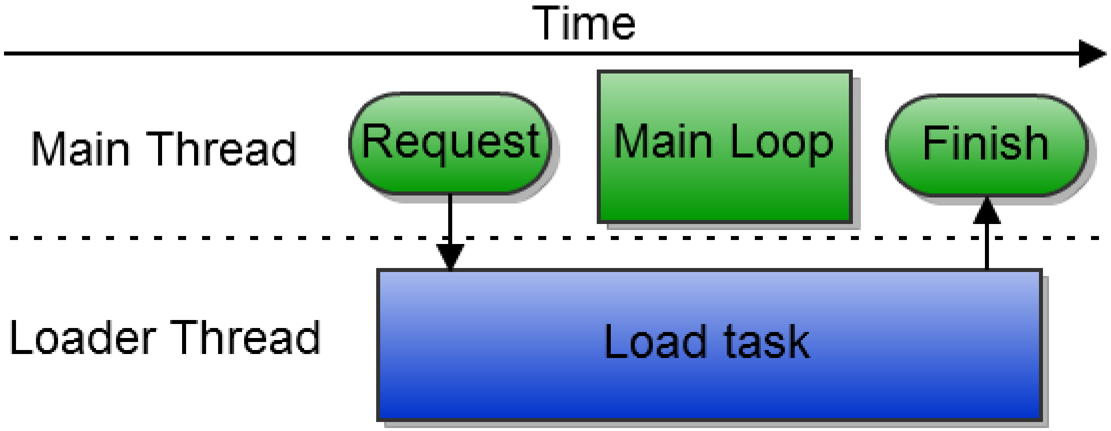
- Disk vs. Streaming
  - – Loading an asset from disk takes milliseconds
  - – Streaming an asset takes 1000+ times longer
  - – Never loading just one…
- Synchronous = locked-up
  - – Can't continue until asset finishes loading

```
,0x8016A950,0x00000001,0x00000085)
HANDLED*** Address 8016a950 has base at 80100000
.6.2 irql:1f  SYSVER 0xf0000565

Name                Dll Base DateStmp - Name
ntoskrnl.exe        80010000 33247f88   al.dll
atapi.sys           80007000 3324804    SIPORT.
Disk.sys            801db000 336015     ASS2.SY
Ntfs.sys            80237000 344eeb4    wvid.sy
NTice.sys           f1f48000 31ec6c8d   loppy.SY
Cdrom.SYS           f228c000 31ec6c9    ull.SYS
KSecDD.SYS          f2290000 335        .SYS
win32k.sys          fe0c2000 34         dll
Cdfs.SYS            fdca2000 3
nbf.sys             fdc35000
netbt.sys           f1f68000
afd.sys             f2008000
Parport.SYS         fdc1400
                    f1dd000
```

Getting the Assets

- Asynchronous Loading



  - A MUST for streaming
  - Most complicated part of streaming system
  - It's a Discipline

Getting the Assets: HTTP

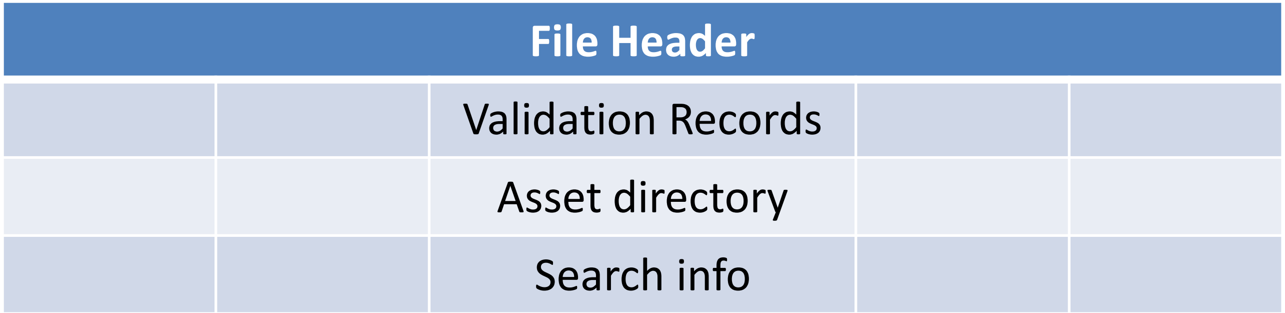
- Well-known text protocol over TCP
- EQII Stats:
  - Proprietary HTTP library
  - 9 parallel connections (variable)
  - 64KB new-connection bias
  - Priority based
  - Software low-bandwidth testing

EQII's Priority

- Highest priority reserved for Synchronous
  - Dedicated HTTP connection as well
- Higher priority for near-camera items
- Very low priority for background downloads
- Ability to change priority

Storing the Assets

- PAK Library (VPL) file format (1 file)



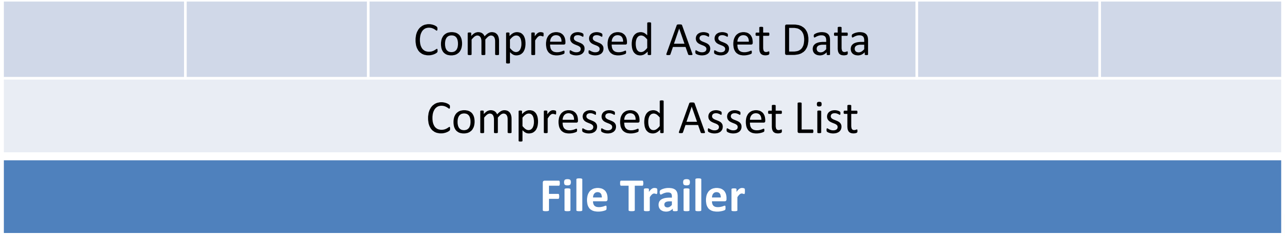
File Header

Validation Records

Asset directory

Search info

- PAK (VPK) file format (1000+ files)



Compressed Asset Data

Compressed Asset List

File Trailer

Storing the Assets

- Guidelines for new file format:
  - Fast to update
  - Minimal changes to update
  - Growable
  - Manage asset changes reasonably
  - Shared among multiple running game instances
    - Very difficult!

Storing the Assets

- Asset Database (VDB) Version 0
  - Multiple files growable to 2GB (identical format)
  - Files could contain up to 512K entries (theoretical)
  - Open Addressing hashtable
  - Free List
  - Unique per game instance (no sharing)

Open Addressing

- Suited for contiguous memory (i.e. files)

File 0					
		File 8	File 44		File 11
<DEL>	File 47				
		File 26		File 28	

- Start = FileHash mod NumSlots
- Index+1 until found or empty (or rollaround)

Storing the Assets

- Asset Database (VDB) Version 0 Format:

<table>
<tr><th colspan="5">File Header</th></tr>
<tr><td></td><td></td><td>Hashtable entries</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Compressed File Data</td><td></td><td></td></tr>
</table>

- Hashtable: 64K entries initially, 512K max
  - Relocate File Data
  - Remove <DEL> entries
  - Recompute new hash locations

Storing the Assets

- Version 0 Drawbacks:
  - Must search each file’s hashtable
  - No sharing across processes
    - Multiple copies of same assets
  - Inefficient file/mutex locking
    - Single-threaded access

Storing the Assets

QUALITY CONTROL

- Testing Considerations
  - Size limit (EQII uses 2GB limit)
  - Potentially very large files
  - Each file created differently
    - Issues are difficult to track without having the file
  - Unit test!
    - Create test cases for all code paths
    - Hard to find/diagnose issues: corruption, load failures

Storing the Assets

- Asset Database (VDB) Version 1 Improvements
  - Sharing!
  - Multithread support
  - Single writer/multiple reader
  - Memory-mapped hashtable (separate file)

Asset Database V1 File Format

- Master file (single)

<table>
<tr><th colspan="5">File Header</th></tr>
<tr><td></td><td></td><td>Hashtable entries</td><td></td><td></td></tr>
</table>

- Data files (multiple)

<table>
<tr><th colspan="5">Minimal File Header</th></tr>
<tr><td></td><td></td><td>Compressed File Data</td><td></td><td></td></tr>
</table>

Sharing Considerations

- Shared Free List
  - Sorted arrays in shared memory
- System-wide Read/Write Mutex
  - 2 Named Semaphores, a Named Mutex, Shared Memory, Thread-Local storage
  - And a Partridge in a pear tree.

Sharing Considerations

- Hashtable reloading
  - Another process can change hashtable
  - Detect change and reload
- Security
  - Global shared memory on Win7/Vista
    - ERROR_ACCESS_DENIED if UAC is active

Optimizations

- Background downloading
- Pre-built Asset Packs
- Bundles

Background Downloading

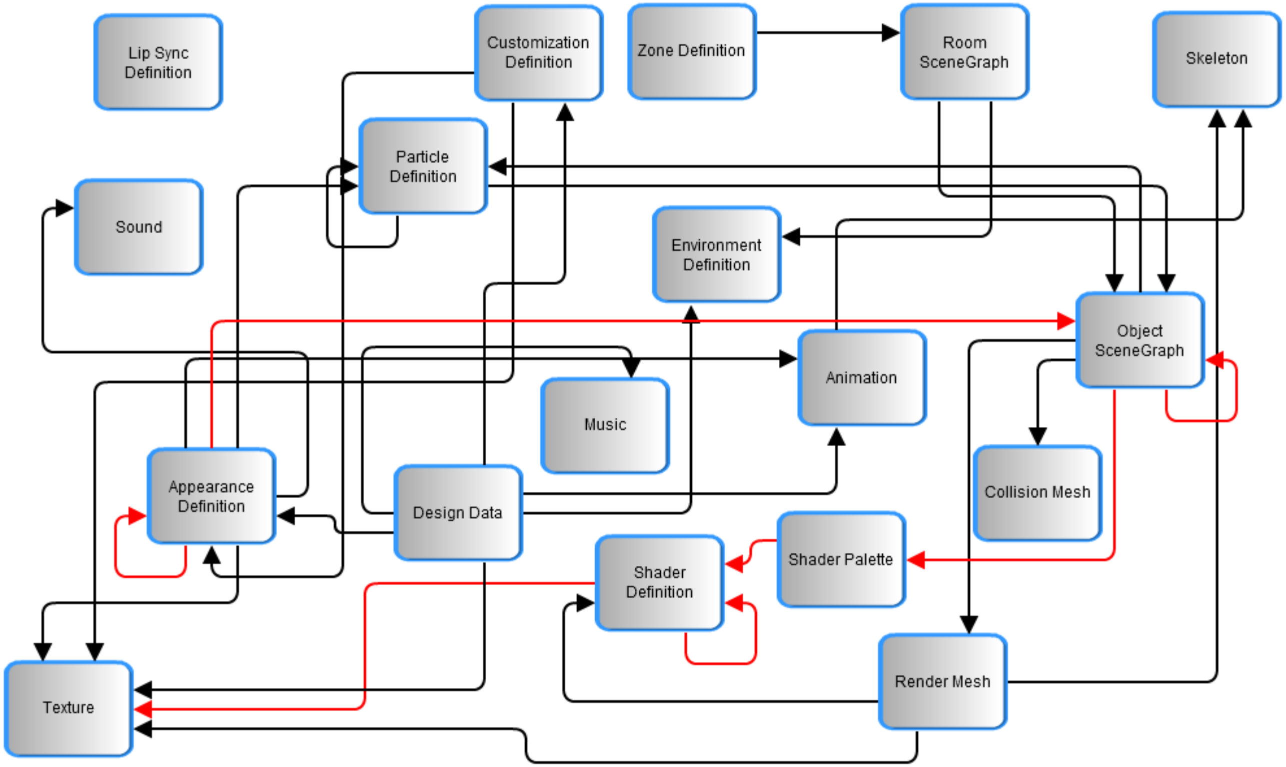
- Entire game?
  - – File list is present in the manifest
  - – Some optimization considerations
    - File order
- Zone-based
  - – Download assets for adjacent zones
  - – Other related zones
  - – Beware CPU cost

Background Downloading

- Asset Lists
  - Files that describe related assets
  - Maintenance?
- Other Users
  - Client uploads list of assets
  - Other clients can use list
  - CDN solution: requires game server
  - Custom server solution: easier

Pre-built Asset Packs

- Some asset types reference other types:



Pre-built Asset Packs

- Additive download time
- Small files have higher download cost
- Assets everyone needs
- Starting areas

Pre-built Asset Packs

- Considerations for delta patching
  - – Distribute assets among several pack files
  - – Asset files in same pack file each build
  - – Asset files in the same order

Bundles

- Groups of related assets
- Streamed down instead of individual assets
- May duplicate assets
- Possibly improve download/loading time

Conclusions

- Mixed Player Response:
  - “I think this is a brilliant move!”
  - “Great! Allows you to start playing in minutes!”
  - “I’d rather download the full game… EQ2 hung on starting.”
  - “I get kicked from groups for not zoning fast”
  - “Created a character in 5 minutes!”

Conclusions

- Lessons Learned
  - Small idea; ~~Big~~ HUGE project
  - Needs wide testing exposure
  - Changing data formats might be worth it.
  - Much better Trial stats

Hiring!

Senior UI Programmer
Senior AI Programmer
Senior Client Programmer
Senior Tools Programmer

Brand New EverQuest Title!

Email résumés to: **jkriegshauser@soe.sony.com**

Questions?

http://www.everquest2.com/