RapidFire: The Easy Route To Low Latency Cloud Gaming Solutions
**AMD RapidFire Technology**

- Most cloud gaming solutions are CPU based
- AMD RapidFire is dedicated cloud hardware and software solution with an API to simplify integration
  - Deliver more HD games streams per GPU with low latency -> 6 x HD 720p30 fps
  - Leverage AMD hardware on both server and client

<table>
<thead>
<tr>
<th>Low Latency</th>
<th>HD Image Quality</th>
<th>Multiple Streams</th>
<th>Virtualization Enablement</th>
</tr>
</thead>
</table>


AMD RapidFire Technology

- Design for many use cases and workflows

- High resolution
- Collaborative
- Virtual Desktop
- Adaptive to Network environment
AMD RapidFire Technology

- 4 independent components using software and hardware acceleration

- **Server**
  - HW H264 Encoding
  - GPU Encoding
  - CPU Encoding
  - Plugins Encoding
  - Multi-stream
  - Multi-API (DX9/11/OpenGL/OpenCL)
  - Encryption
  - Desktop and window direct encoding

- **Network**
  - Adaptive protocol

- **Client**
  - HW H264 Decoding
  - CPU Decoding
  - Custom GPU Decoding
  - Multi-platform (Win/Linux/IOS/Android)
  - Multi-API (DX9/11/OpenGL/OpenCL)

- **User Interface**
  - Touch
  - Keyboard
  - Mouse
  - Multi-platform
GCN | Render Graphics
Frame Grab | Low-latency frame capture
VCE | HD multi-stream Compression
AMD RapidFire

Server
- CPU
- H264
- Encode
- System Memory
- RAPIDFIRE API
- DRIVER
- GPU
- Frame Buffer
- HW
- H264
- Encode
- Keyboard
- Mouse events
- Network

Client
- Keyboard
- Mouse
- decode
- Network

Frame Buffer
- Frame Buffer
- HW
- H264
- Encode
- Keyboard
- Mouse
AMD RAPIDFIRE TECHNOLOGY
DATA FLOW OVERVIEW

Server
- GAME SERVER
- Radeon Sky GPU
  - 3D ENGINE
  - ENCODE
  - FRAME BUFFER
- NETWORK

Client
- NETWORK
- DECODE
- UI
- FRAME BUFFER

RF API module
external module
**AMD RAPIDFIRE TECHNOLOGY**

**SERVER SIDE DATA FLOW**

- **Network component** transfers UI events from the client to the server.
AMD RAPIDFIRE TECHNOLOGY

SERVER SIDE DATA FLOW

- Network component transfers UI events from the client to the server
- Game server sends commands to GPU to draw next frame
AMD RAPIDFIRE TECHNOLOGY

SERVER SIDE DATA FLOW

- Network component transfers UI events from the client to the server
- Game server sends commands to GPU to draw next frame
- GPU distributes the work among execution resources and produces resulting frame into the frame buffer
AMD RAPIDFIRE TECHNOLOGY
SERVER SIDE DATA FLOW

- Network component transfers UI events from the client to the server
- Game server sends commands to GPU to draw next frame
- GPU distributes the work among execution resources and produces resulting frame into the frame buffer
- VCE is performing asynchronous frame sequence encoding into H264 video stream and the data is fetched to system memory by the app
AMD RAPIDFIRE TECHNOLOGY
SERVER SIDE DATA FLOW

- Network component transfers UI events from the client to the server
- Game server sends commands to GPU to draw next frame
- GPU distributes the work among execution resources and produces resulting frame into the frame buffer
- VCE is performing asynchronous frame sequence encoding into H264 video stream and the data is fetched to system memory by the app
- SW-encoded audio stream is merged with the video stream and sent to the network
AMD RAPIDFIRE TECHNOLOGY

CLIENT SIDE DATA FLOW

- Game client receives H264 stream from the server using RTSP protocol
Game client receives H264 stream from the server using RTSP protocol.

The client sends the stream to AMD GPU which performs H264 hardware decoding to the frame buffer.
Game client receives H264 stream from the server using RTSP protocol.

The client sends the stream to Radeon GPU which performs H264 hardware decoding to the frame buffer.

Sound stream is decoded using software audio codec and sent to audio hardware.
AMD RAPIDFIRE TECHNOLOGY

CLIENT SIDE DATA FLOW

- Game client receives H264 stream from the server using RTSP protocol
- The client sends the stream to Radeon GPU which performs H264 hardware decoding to the frame buffer
- Sound stream is decoded using software audio codec and sent to audio hardware
- UI events are collected by the client and sent over the network to the server

Diagram:
- Game client
- Radeon GPU
- Frame buffer
- Audio device
- Input devices
- Network
- Decode audio
- UI
AMD RAPIDFIRE API

COMPONENTS

Server component
The server component provides functions for the:
- Encoding of video and audio data
- Color space conversion
- Capturing of the desktop
- Handling of multiple render targets
- Interoperability with OpenGL, D3D9 and D3D11

Client component
The client component provides functions for:
- Decoding of video and audio streams
- Color space conversion
- Interoperability with OpenGL, D3D9 and D3D11

Network component
The Network component is a sample implementation of video and audio streaming based on the LIVE555 Media Server.

User Interface component
The UI component provides functions to:
- Capture user events on the client
- Send the events to the server for processing
AMD RAPIDFIRE API
THE SERVER COMPONENT

Initialization

- Create RenderTarget
- rfCreateEncodeSession
- rfRegisterRenderTarget
- rfCreateEncoder

Render Loop

- Draw to free RenderTarget
- rfEncodeFrame
- rfGetEncodedFrame
- rfRtspServerSendFrame
- Done?
rfCreateEncodeSession
- Creates an encoding session on the server. The sessions encapsulates the following components:
  - Rendering context/device
  - Compute context that is used for the color space conversion
  - Render targets
  - Desktop
  - The encoder: SW, VCE or IDENTITY

- The following session types are supported
  - OpenGL
  - DX9 / DX9Ex
  - DX11
  - Desktop capturing
**Initialization**

- **rfCreateEncodeSession**
  - Creating an OpenGL session that uses the VCE HW encoding

  ```
  RFProperties props[] = { RF_GL_GRAPHICS_CTX, (RFProperties)hGLRC, 
  RF_GLDEVICE_CTX, (RFProperties)hDC, 
  RF_ENCODER, (RFProperties)RF_VCE, 
  0 };
  
  // Create RapidFire encoding session 
  RFEncodeSession session = rfCreateEncodeSession(props);
  ```

- **rfRegisterRenderTarget**
  - Input: Name of the OpenGL Texture, dimension of the texture
  - Output: The index used by RF to identify this render target

  ```
  for (unsigned int i = 0; i < NUM_RENDER_TARGETS; i++)
  {
    rfRegisterRenderTarget(session, RF_RT_GL_TEXTURE, uiTexName[i], uiWidth, uiHeight, &renderTargetIdx[i]);
  }
  ```
AMD RAPIDFIRE API
THE SERVER COMPONENT

Initialization

Create RenderTarget

rfCreateEncodeSession

rfRegisterRenderTarget

rfCreateEncoder

- Creating an Encoder using a preset configuration

rfCreateEncoder(session, uiWidth, uiHeight, RF_ENCODE_FAST)

- The following presets are supported:
  - RF_ENCODE_FAST
  - RF_ENCODE_BALANCED
  - RF_ENCODE_QUALITY

rfCreateEncoder2

- Creating an encoder using properties

```c
// Create encoder using properties
RFProperties props[] = { RF_ENCODER_PROFILE, (RFProperties)RF_MAIN,
RF_ENCODER_LEVEL,   (RFProperties)41,
RF_ENCODER_BITRATE, (RFProperties)6000000,
RF_ENCODER_FPS,     (RFProperties)30,
0 };

rfCreateEncoder2(session, uiWidth, uiHeight, props);
```
AMD RAPIDFIRE API

THE SERVER COMPONENT

- **rfEncodeFrame**

  
  ```
  rfEncodeFrame(session, renderTargetIdx[uiCurrentRT]);
  ```

  - Non-blocking call to submit a frame for encoding

- **rfGetEncodedFrame**

  ```
  // Check if encoded frame is ready
  if (rfIsEncodedFrameReady(session))
  {
  
  if (rfGetEncodedFrame(session, &uiBitStreamSize, (void**)&pBitStreamdata) == RF_STATUS_OK)
  {
  
  if (uiBitStreamSize > 0)
  {
  
  // Send encoded frame to Network
  rtspStatus = rfRtspServerSendFrame(rtsp_sn, pBitStreamdata, uiBitStreamSize, tv.tv_sec, tv.tv_usec, 1);
  
  
  }
  
  }
  
  }
  ```

  - Check first if a frame is ready
  - If a frame is available get the data to system memory
  - Send the frame over the network to the client
AMD RAPIDFIRE API
THE CLIENT COMPONENT

Initialization

Create Target Textures
rfCreateDecodeSession
rfRegisterTargetTexture
rfCreateVideoDecoder

Render Loop

Wait to receive frame
rfDecodeFrame
Display frame using Target Texture
Done?
AMD RAPIDFIRE API
THE CLIENT COMPONENT

Initialization

rfCreateDecodeSession
- Creates a decoding session on the client. The sessions encapsulates the following components:
  - Rendering context/device
  - Compute context that is used for the color space conversion
  - Target Textures
  - Decoder: SW or UVD

rfRegisterTargetTexture

rfCreateVideoDecoder

Create Target Textures
Create Target Textures
rfCreateDecodeSession
rfRegisterTargetTexture
rfCreateVideoDecoder

Initialization

**rfCreateDecodeSession**

```
RFProperties props[] = { RF_GL_GRAPHICS_CTX, (RFProperties)hGLRC,
                        RF_GL_DEVICE_CTX,   (RFProperties)hDC,
                        RF_DECODER,         (RFProperties)RF_UVD,
                        0                   };

// Create RapidFire decoding session
RFDecSession session = rfCreateDecodeSession(props);
```

- Creates an OpenGL session that uses the UVD decoder

**rfRegisterTargetTexture**

```
rfRegisterTargetTexture(session, uiTextureName);
```

- Registers an OpenGL texture
- The texture will be used to store the decoded frame
AMD RAPIDFIRE API
THE CLIENT COMPONENT

Initialization

rfCreateVideoDecoder

- Create the actual decoder

rfCreateVideoDecoder(session, uiWidth, uiHeight);
AMD RAPIDFIRE API
THE CLIENT COMPONENT

rfDecodeFrame

- Decodes a frame
- The decoded frame is stored in the registered texture
- Now the application can use the texture to display the frame

```
rfDecodeFrame(session);
```
We have taken a brief look at the API
- Detailed API specifications, sample code available in the SDK
  - Illustrates how to implement alternative encode/decode/network/etc. for non-AMD platforms

Are developers currently using the API?
- Yes, let’s take a look at some implementations...
Swiich solution by Eureva

Philippe Martineau – philippe.martineau@eureva.fr
WHY CLOUD GAMING?

**PUBLISHERS**
- Majors
- Independents
- Studios

**DISTRIBUTORS**
- TV channels
- Telcos
- Online content

**MANUFACTURERS**
- Consoles & micro consoles
- Tablets
- Connected TVs

**USERS**
- Casuals
- Gamers

- No piracy
- New distribution channel
- Homogeneous platform
- New content offerings
- Accessible PC catalog
- Multi-screen
- Upward compatibility
- Contents
- Instant access
- Try & Buy
- Multi-screen
- Affordable hardware
Make videogames accessible from the cloud on all screens

Current implementation: virtualization of any DirectX 9, 11 or OpenGL application
SWIICH DEMO

🔥 Crisis 3 – virtualized & streamed real-time

- AMD FirePro Graphics
- Crisis 3 integration without any modification: "zero touch"
- Seamless game execution & AMF encoding on AMD FirePro
- 720p image
- Bandwidth compatible with existing networks (<5Mbps)
- Client: any video-capable terminal
- Very low latency
PERFORMANCE ACHIEVEMENTS

- AMD FirePro Graphics
- 3D application running & encoding
  - 40-60 frames per second
- Image encoding and capture: 8 to 16ms
- Image decoding on low-end hardware < 16ms
- Image display on low-end hardware < 16ms

**Overall LAN roundtrip:** ~40ms (Joystick input from Client to Server; Image capture, compression streaming and display)

**Full WAN roundtrip latency:** ~100ms on real internet networks
CLOUD-GAMING USE-CASES

**Cloud-based streaming**
- Streaming 3D apps from the Cloud, and transitioning from a licensing model to SaaS
- Instant 3D software demos from the Cloud
  - Explore software potential, with no client download; graphic card requirements are provided by the cloud
- Embed in-game video-advertising
- Built-in multiple screen sharing for gaming events or simply to watch your friends on the cloud
- Game developments with graphics provisioned on-demand from servers

**Point-to-point streaming**
- Graphic card to device streaming for gaming scenarios
  - Basic game-screen sharing on remote device: tablet, phone...
  - SDK to access in-game streamed content (i.e. specific gameplay on a cell-phone or tablet)

**Ultra-high definition interactive screens for gaming events (4k and beyond images)**

*Streaming: Whenever pixels are on networks...*
Leap Computing

Direct→Game

Alexander Nataros, CEO
The Importance of RapidFire, and What Counts for LEAP

- Low Latency Frame Grab
- Low Latency Encoding
- 1080p & 720p Support at 30 or 60 FPS
- Support for all major mobile client platforms
- Low CPU demand due to OpenCL and Rapidfire
- AAA Gaming Titles Supported, All Genres
- Incredible User Density per GPU
The Direct-Game Engine

- No development cycle for Game Developers
- Efficient Operations for integrating new game content
- DX11, DX9, OpenGL, and Mantle Fully Supported
- Automated game deployment and density management
- Software redundancy to ensure smooth gameplay
- Fast, efficient encoders and decoders for streaming
- Intelligent, scaling encoders that only do what’s needed
- Radeon Sky complete line fully supported
- 99.999% System Uptime
- Deployable TODAY
LEAP COMPUTING, INC. – CLOUD GRAPHICS ACCELERATION
LEAP DIRECT GAME PLATFORM – CLOUD GAMING WITH RAPIDFIRE

RapidFire and OpenCL – Density and Efficiency

- <1mbps for 720p
- <2mbps for 1080p
- <20ms “Preflight” encoding
- Encode frames without Leaving GPU via VCE and OpenCL
- Stream only what’s needed

[Diagram showing the process of rendering, grabbing, encoding, and streaming frames via Rapidfire]

Render Virtualized Frame (Direct-Game) → Rapidfire Frame Grab → Leap Encode Engine → Stream Encoded frame via Rapidfire
GPU Density, Radeon Sky, and what’s important.

Traditional Encoding

Encode
GPU Density, Radeon Sky, and what’s important.

- Elastic Encoding

- Purchased Encode

- Free Encode
LEAP COMPUTING, INC. – CLOUD GRAPHICS ACCELERATION
LEAP DIRECT GAME PLATFORM – CLOUD GAMING WITH RAPIDFIRE

GPU Density, Radeon Sky, and what’s important.

- Rapidfire standardizes cloud gaming industry expectations
- AMD Radeon Sky allows for many concurrent instances of advanced gaming and efficient encoding
- Consistent hardware and software sources ensure quality and performance in deployments
- RapidFire provides quick, easy integration to powerful tools for cloud integration

LIVE DEMOS AT AMD BOOTH 1024!
LeapComputing.com
CONCLUSION

- RapidFire provides a cross platform framework for cloud gaming
  - Already being used by 3rd parties to implement remoting solutions

- Stop by the AMD booth 1024 in the expo to see these solutions

- If you would like to get access or more information on RapidFire technology, email requests to FirePro.Developers@amd.com
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

© 2013 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.