Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Mobile Cloud Gaming

KYUNGMIN LEE, DAVID CHU, EDUARDO QUERVO, JOHANNES KOPF, YURY DEGTYAREV, SERGEY GRIZAN, ALEC WOLMAN, JASON FLINN

UNIVERSITY OF MICHIGAN, MICROSOFT RESEARCH, ST. PETERSBERG POLYTECHNIC UNIVERSITY, SIBERIAN FEDERAL UNIVERSITY

MOBI’SYS ’15, MAY 18-22, 2015
Background: Cloud Gaming

- Processing and rendering done on cloud
- Client sends inputs, receives rendered images

Benefits
- Better graphics – use server’s processing hardware
- Easy to develop – no compatibility issues
The Problem

- Lacks *real-time interactivity*
  - High latency sensitivity affects gameplay
  - Buffering impossible due to changing user input

**Fig. 1 (a) – Standard Cloud Gaming: Frame time depends on net latency**
Solution

- Speculate frames until next response
- Challenges – dynamism and sensitivity

Fig. 1 (b) – Outatime: Frame time is negligible
Fig. 2: Outatime Architecture
Speculation for Navigation

- Create discrete time Markov Chain
  - Define input $N_t = \{\delta_{x,t}, \delta_{y,t}, \delta_{z,t}, \theta_{x,t}, \theta_{y,t}, \theta_{z,t}\}$
  - Given input $n_t$, find most likely input for next frame $\hat{N}_{t+1}$
  - For RTT $\lambda$:
    \[
    \hat{N}_{t+\lambda} = \arg\max_p (N_{t+1} | N_t = n_t) \times \prod_{i=1}^{\lambda-1} p(N_{t+i+1} | N_{t+i})
    \]
- Also track error estimate
Speculation for Navigation

- Supersampling – Collect data as fast as input device allows
  - Improves accuracy
  - Reduces sampling noise

- Prediction accuracy improves over 5 min. of samples

- Use training data of other players
  - Characteristics depend on skill level

Fig. 5 – Error distribution for different training periods
Speculation for Navigation

- Video Shake
  - Caused by small prediction errors at low-latency
  - Fixed using Kalman Filtering

- Kalman Filter
  - Emphasizes measured values for low RTT (< 40ms.)
  - Emphasizes predicted values for high RTT (> 40ms.)
Misprediction Compensation

- Image-based Rendering
  - Transform rendered prediction to be more accurate

- Clipped Cube Map
  - Render areas surrounding frame in case they are needed
  - Limit size based on expected error values
Clipped Cube Map

Fig. 7 – Cube Map Example
Image-based Rendering (IBR)

Figure 6 – Image-based Rendering in Fable 3
Speculation for Impulse Events

- Much harder to predict
- Solution: speculate different possibilities in parallel
- Create speculative input sequence
- As RTT increases, speculative sequence space grows exponentially
- Two methods to decrease speculative sequence size
  - Subsampling
  - Time-Shifting
Subsampling and Time-Shifting

- **Subsampling**
  - Sample inputs at a period $\sigma > 1$ clock tick
  - Reduces state space to $2^{\frac{\lambda}{\sigma}}$
  - On its own, likely to miss samples

- **Time-shifting**
  - Shift every input activation to occur on the nearest subsample
  - Solves problem of subsampling
  - Can shift inputs backwards since state is speculative
Figure 10: Subsampling and time-shifting impulse events allows the server to bound speculation to a maximum of four sequences even for RTT = 256ms. Screenshots (b) – (e) show speculative frames corresponding to four activation sequences of weapon fire and no fire.
Speculation for Impulse Events

- Not all inputs are binary
  - Alternate firing for a weapon
- State space grows quickly (e.g. $3^\lambda$ instead of $2^\lambda$)
- Outatime supports ternary and quaternary events for RTT $\leq 128$ms

- Some impulse events delay tolerant
  - Do not speculate
  - Instead, use time compression to account for RTT delay
Checkpoint and Rollback

- Supports page-level and object-level checkpointing
  - Depends on density of Simulation State Objects (SSOs)

- Page-level checkpointing
  - Copy page on page write
  - Invalidate mis-speculated data
  - Copy back on rollback

- Object-level checkpointing
  - Use inverse functions when rolling back
Implementation

- Manually modified Doom 3 code
- Doom 3 master with multiple speculative slave versions
  - render
  - undo
  - commit
  - rendercube
- Used hardware to improve compression and video encoding
Experiment

- 3 Experiments
  - Doom 3: 23 people
  - Doom 3: 18 gamers
  - Fable 3: 23 people

- Measured on 3 metrics:
  - Mean Opinion Score
  - Skill Impact
  - Task Completion Time
Results

- Minor decrease in quality for latencies up to 128 ms
- More noticed by gamers
  - Larger/faster movements cause greater mispredictions
  - May be more sensitive as a player to such effects
- Significant reduction in skills at higher latencies
- Task completion relatively unaffected
  - Improvement over regular cloud gaming
Results

Figure 11: Impact of Latency on User Experience

Figure 12: Remaining Health

Figure 13: Task Completion Time
Performance

- Bandwidth 1.97x higher than standard cloud gaming
  - 1.04 Mbps at RTT = 128ms
- Framerate = 52fps at 95th percentile
Strengths and Weaknesses

**Strengths**
- Many useful and practical tactics to minimize bandwidth
- Effective and varied predictive measures are taken for different classifications of inputs
- Provides foundation to make cloud gaming practical with relatively low latency

**Weaknesses**
- Does not seem scalable to games with many inputs or fast inputs (e.g. RTS)
- Requires significant code restructuring
  - Harder to use on existing games
- Does not consider faster framerates
  - Modern games are frequently 60Hz
Reference

Questions?

Image Source: http://i.ytimg.com/vi/gCSmykwODqA/maxresdefault.jpg
Figure 3: Doom 3 Navigation Prediction Summary. Roll ($\theta_z$) is not an input in Doom 3 and need not be predicted.

Figure 4: Prediction for Yaw ($\theta_x$), the navigation component with the highest variance. Error under 4° is imperceptible.
Figure 8: Angular coverage of 99% of prediction errors is much less than 360° even for high RTT.

Figure 9: Misprediction’s visual artifacts appear as smears which we mitigate.