# A Benchmark to Evaluate Mobile Video Upload to Cloud Infrastructures

Afsin Akdogan, Hien To, **Seon Ho Kim** and Cyrus Shahabi

Integrated Media Systems Center

University of Southern California, Los Angeles, CA USA

09/05/2014

5th BPOE Workshop at VLDB

# Outline

- **Motivation**
- MediaQ
- Benchmark Design
- Experimental Evaluation
- Conclusion and Future Directions

# Motivation

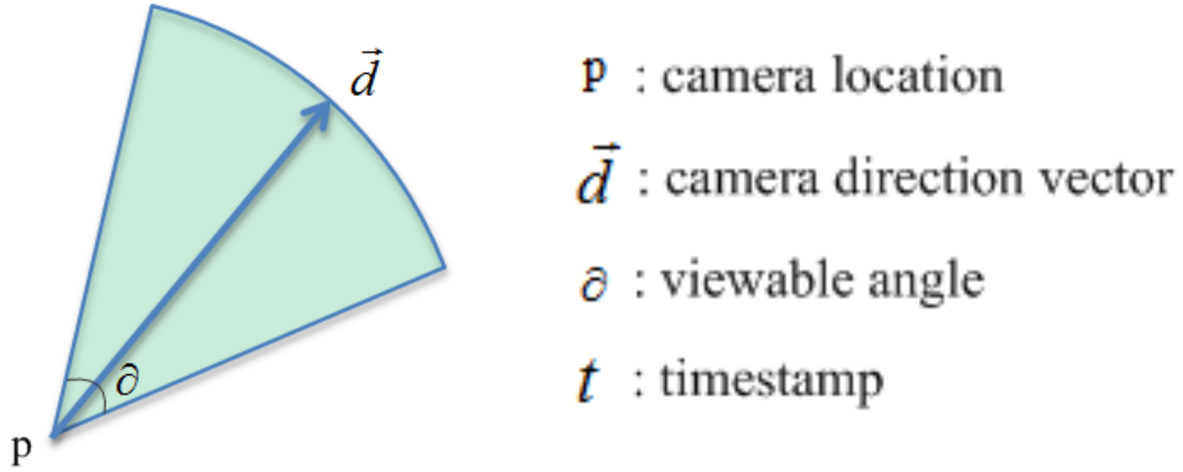
- <u>Mobile Revolution:</u>
  - Smart phones, tablets and wearable technologies




  - There will be over **10 billion** mobile devices by 2018.
  - 54% of them smart devices. Up from 21% in 2013.

# Motivation

- In line with the increase in mobile devices the amount of mobile video rapidly grows: more cloud-based mobile media applications
- Recent trend in mobile media: not just contents but also geospatial metadata
  - For better archiving and searching
- <u>Geospatial metadata</u>
  - Combines files with metadata extracted from the video using sensors available in the devices
    - camera location
    - camera direction
    - viewable angle
    - ....



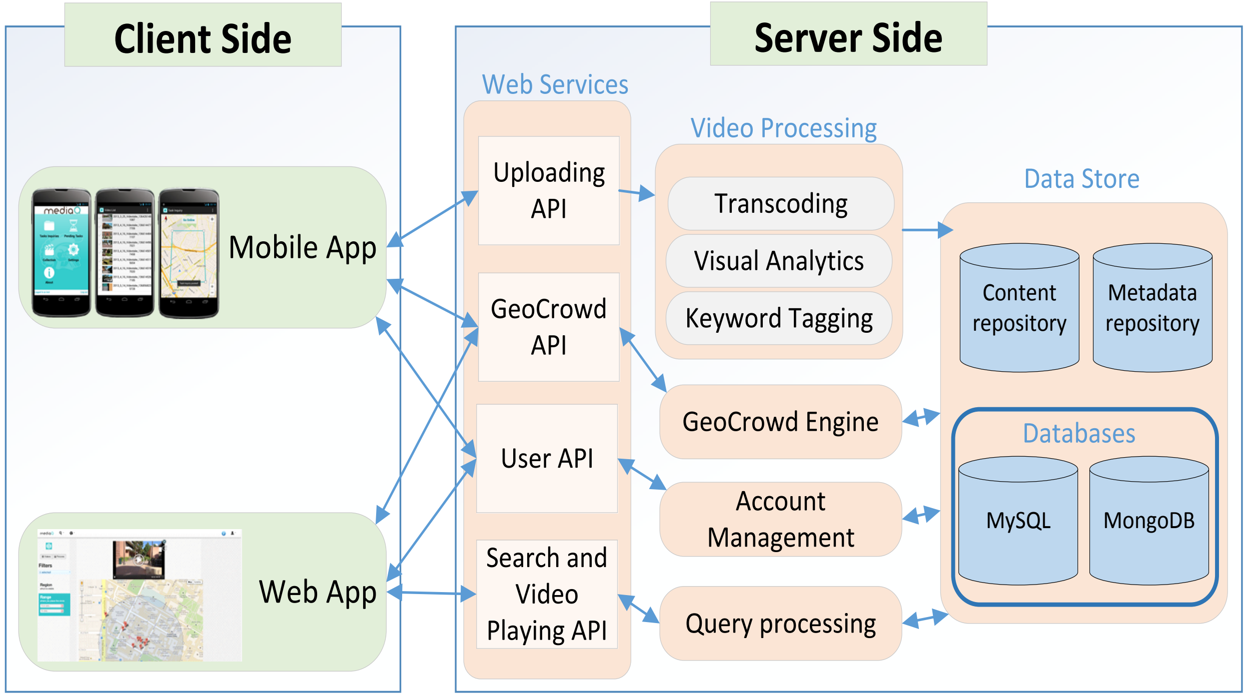
# Motivation

- <u>Mobile Video</u>
  - Metadata enables advanced querying features.
- <u>Future of Mobile Video:</u>
  - It will increase 14-fold by 2018, accounting for **69%** of total mobile data traffic.
- Cloud-based mobile media applications
- How well can cloud support such applications?

- Motivation
- **MediaQ**
- Benchmark Design
- Experimental Evaluation
- Conclusion and Future Directions

# MediaQ

- An example of resource intensive mobile video application on cloud [ACM MMSys 2014]
- Functions: collect, search, and share user-generated mobile videos using automatically tagged geospatial metadata (location, direction, etc.).
- Advanced Querying: point, range, directional, etc.




Architecture

- Motivation
- MediaQ
- **Benchmark Design**
- Experimental Evaluation
- Conclusion and Future Directions

# Benchmark Design

- Goal
  - Identify the appropriate server type for mobile video applications
- Challenge
  - Plenty of server options available (compute optimize, memory optimized, etc.)
  - Price varies considerably. In Azure, the most expensive server is **245** more costly than the cheapest one. What configuration is cost-effective?

| Server Type | Amazon EC2 | | Microsoft Azure | | Google Compute | |
|---|---|---|---|---|---|---|
| | price ($/hour) | | price ($/hour) | | price ($/hour) | |
| | smallest | largest | smallest | largest | smallest | largest |
| General purpose (m) | 0.07 | 0.56 | 0.02 | 0.72 | 0.077 | 1.232 |
| Compute optimized (c) | 0.105 | 1.68 | 2.45 | 4.9 | 0.096 | 0.768 |
| Memory optimized (r) | 0.175 | 2.8 | 0.33 | 1.32 | 0.18 | 1.44 |
| Disk optimized (i) | 0.853 | 6.82 | - | - | - | - |
| Micro (t) | 0.02 | 0.044 | - | - | 0.014 | 0.0385 |

Dollar per hour prices of the smallest and the largest servers at each server groups

# Benchmark Design

- Servers

| Type | Memory | CPU | Disk | Network Bandwidth |
|---|---|---|---|---|
| *m-small* | 3.75 GB | 1 VCPU | 4 GB SSD | no info. |
| *c-small* | 3.75 GB | 2 VCPUs | 32 GB SSD | no info. |
| *r-small* | 15.25 GB | 2 VCPUs | 32 GB SSD | no info. |
| *i-small* | 30.5 GB | 4 vCPUs | 800 GB SSD | no info. |
| *m-large* | 30 GB | 8 VCPU | 160 GB SSD | no info. |
| *c- large* | 60 GB | 32 VCPUs | 640 GB SSD | no info. |
| *r- large* | 244 GB | 32 VCPUs | 2 x 320 GB SSD | 10 Gigabit Ethernet |
| *i- large* | 244 GB | 32 vCPUs | 8x800 GB SSD | 10 Gigabit Ethernet |

# Benchmark Design

- Quantify the performance of cloud for mobile media applications
- Methodology:
  - Break the general video upload into sub components
  - Define a cross-resource metric and compare the performance of all components using the single metric
  - Spot the component(s) that becomes the bottleneck in the workflow and choose the server types accordingly to improve the bottleneck.
    - E.g., compute optimized machines are designed for CPU intensive tasks.

# Benchmark Design

- Video uploading with geospatial metadata
- Video upload **workflow** consisting of three phases:
  1. video transmission (network)
     - Upload video file and metadata extracted from the video from mobile clients to cloud servers
  2. metadata insertion to database
     - Insert metadata into relational database tables (e.g., MySQL)
  3. video transcoding
     - Reduce resolution of the video and change the type if necessary (e.g., from mp4 to avi)

# Benchmark Design

- Metric
  - **Throughput**: *Processed frames per second*

- Throughput in the Workflow
  1. video transmission
     - The number of uploaded frames per second

  2. metadata insertion to database
     - The number of frames inserted into the database

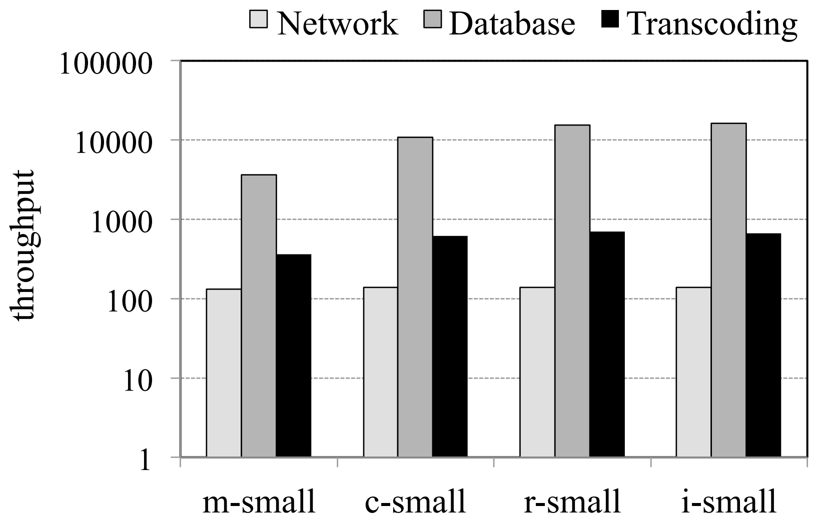
  3. video transcoding
     - The number of frames transcoded

- Motivation
- MediaQ
- Benchmark Design
- **Experimental Evaluation**
- Conclusion and Future Directions

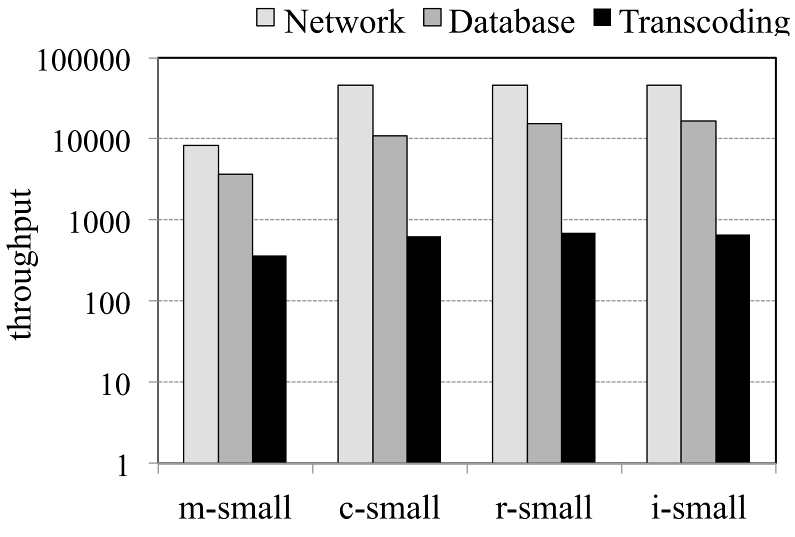
# Experimental Evaluation

- <u>Initial Overall Performance Analysis</u>
  - Used smallest servers in 4 server types on Amazon EC2
  - Enabled *bulk insert* (collect 1K records and insert)
  - Used *ffmpeg* for transcoding, a leading transcoding library
  - Enabled multi-threading for Database insertion and Transcoding
  - <u>Observation:</u> Transcoding becomes a major bottleneck

| | |
|---|---|
| **m** | General purpose |
| **c** | Compute optimized |
| **r** | Memory optimized |
| **i** | Disk optimized |




**a) single video**

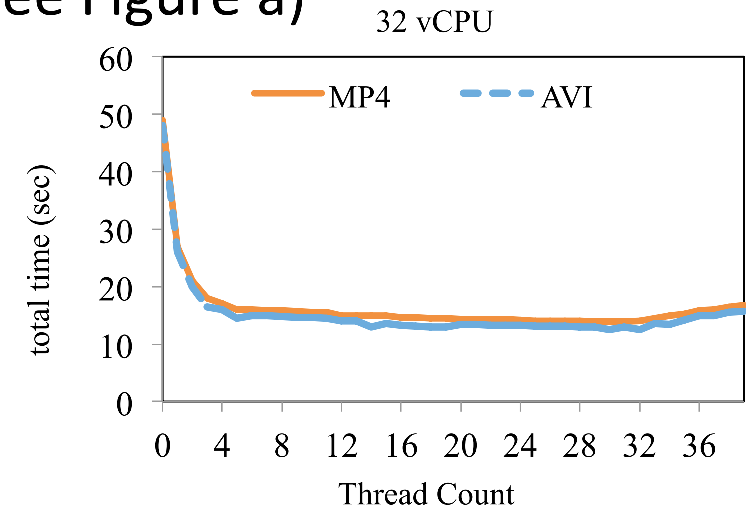



**b) multiple videos**

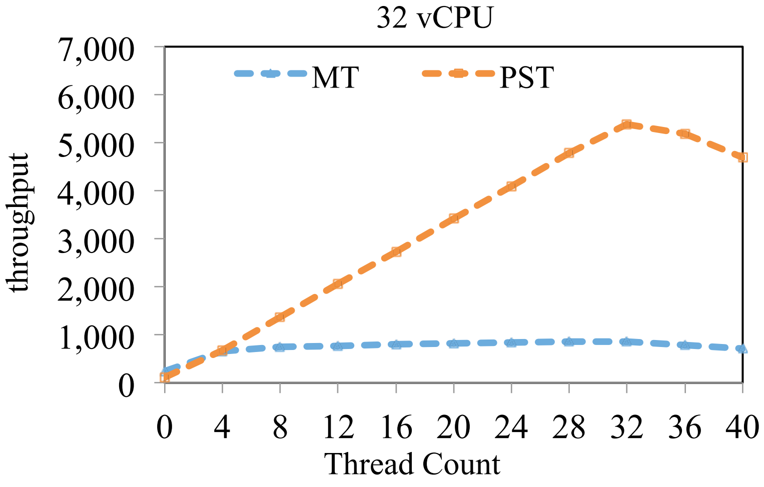
log-scale

# Experimental Evaluation

- <u>Transcoding Performance</u>
  - First, **multi-threading**
    - MT: Enable multi-threading on a single video and transcode it asap
    - PST: Run multiple single-threaded transcoding tasks
  - <u>Observation:</u> ffmpeg does not scale up as the number of threads increases (See Figure a)




**a)** multi-thread ffmpeg on the same video for different video output types




**b)** multi-thread (MT) vs. parallel single-thread (PST) ffmpegs

# Experimental Evaluation

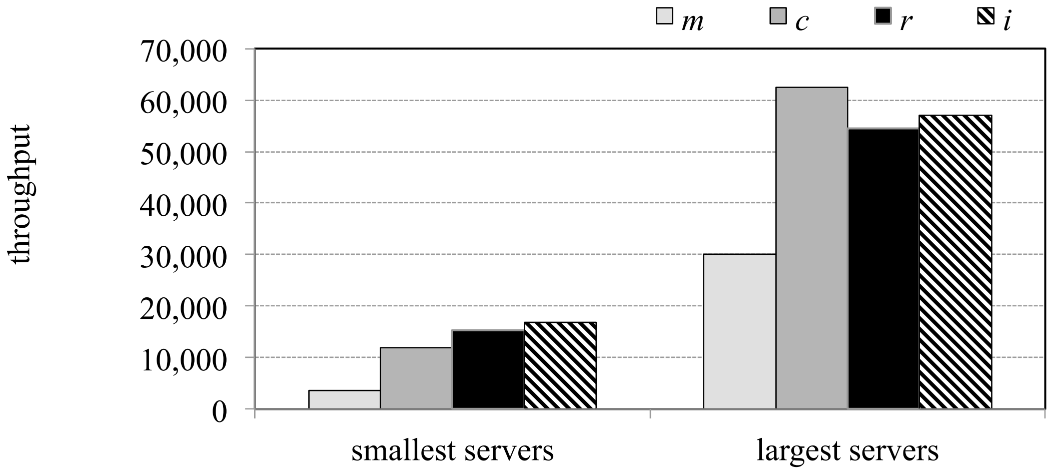
- <u>Transcoding</u>
  - Second, **reducing video quality**
  - Input video resolution is 960x540
  - <u>Observation:</u> Throughput increases as the resolution decreases. However, the percentage improvement **diminishes** when the output video resolution becomes too smaller because loading the input video, frame by frame, is a constant cost which largely contributes to the total transcoding cost.

Cut the size of each dimension by half at each experiment

| Output resolution | MP4 | | AVI | |
|---|---|---|---|---|
| | throughput | % improvement | throughput | % improvement |
| 480x270 | 623 | - | 626 | - |
| 240x136 | 842 | 35% | 839 | 34% |
| 120x68 | 980 | 57% | 982 | 57% |
| 60x34 | 1038 | 66% | 1048 | 67% |

# Experimental Evaluation

- <u>Database</u>
  - Average length of a row is 319 bytes and no index on metadata table
  - We used the smallest and the largest servers in 4 different server types
  - <u>Observation</u>:
    - Metadata insertion is I/O-intensive but disk-optimized machines do not expressively outperform others because video upload is **mostly *append-only***. Disk-optimized instances are tuned to provide fast **random I/O**; however, in *append-only* datasets random access is not much used.
    - With largest servers, multi-threading is enabled and compute-optimized server handles multiple threads better. Therefore, it outperform others.

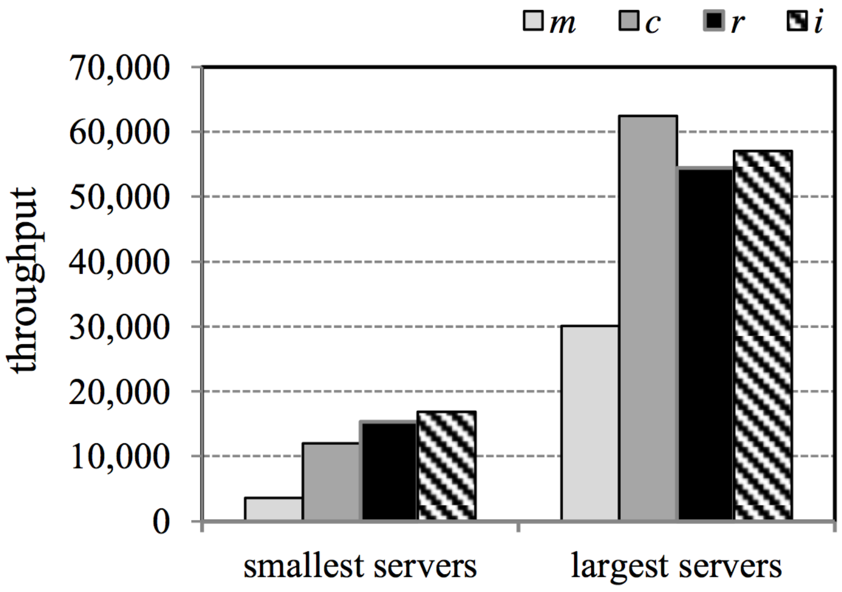



| | |
|---|---|
| **m** | General purpose |
| **c** | Compute optimized |
| **r** | Memory optimized |
| **i** | Disk optimized |

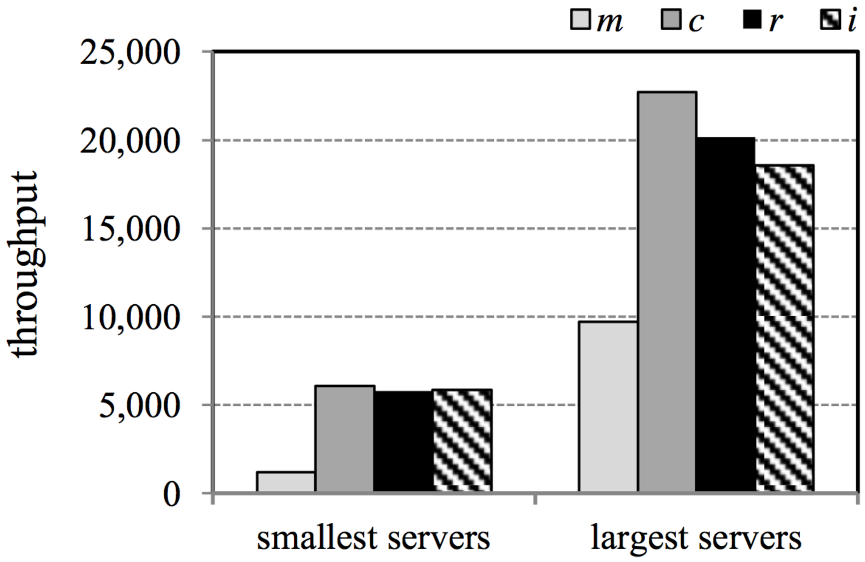
# Experimental Evaluation

Database insertion with index

- B-tree and hash tree



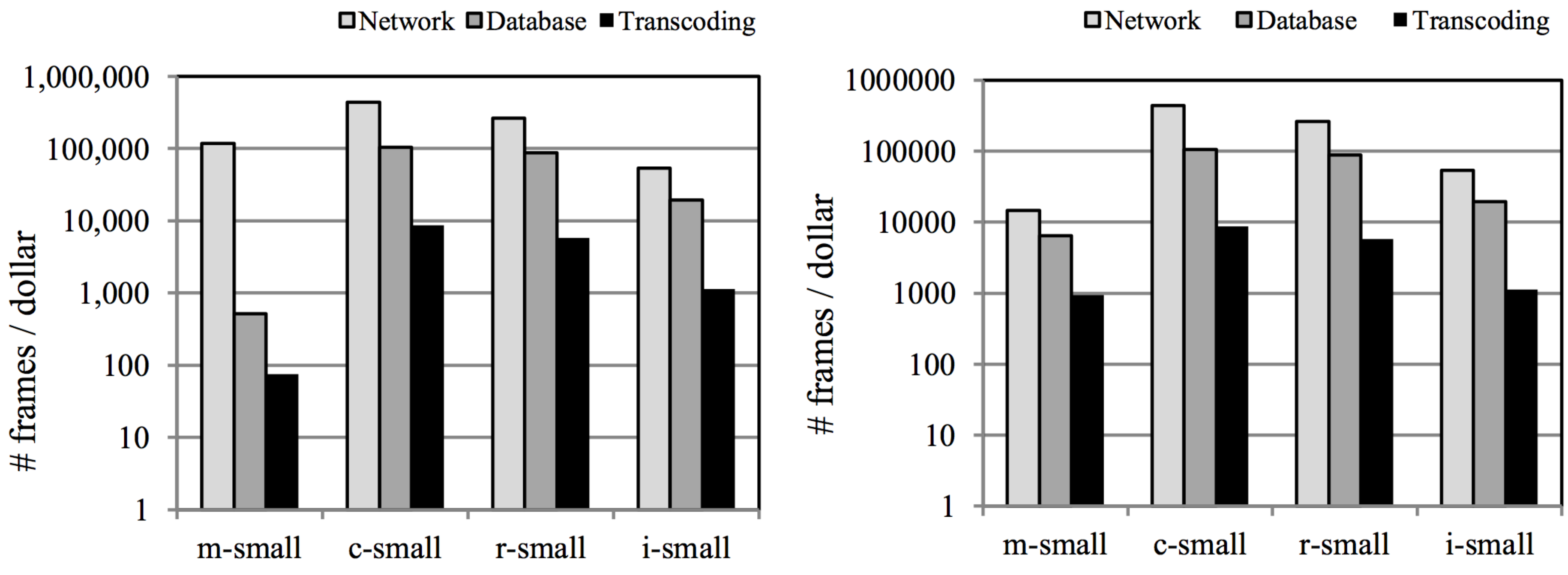

**a)** no index




**b)** with index

# Experimental Evaluation

- Performance-Cost: number of frames per dollar




**a)** 4GB of data **b)** 32GB of data

- Motivation
- MediaQ
- Benchmark Design
- Experimental Evaluation
- **Conclusion and Future Directions**

# Conclusion and Future Directions

- Conclusion
  - Investigated mobile video upload performance in cloud environment
  - Transcoding (a CPU-intensive tasks) is the major bottleneck
  - Compute-optimized servers provides the best performance
  - ffmpeg library does not scale up linearly; therefore, best way to utilize multicore CPUs is to run multiple single-threaded ffmpeg tasks.
- Future Directions
  - Partitioning the dataset and **scaling out** to multiple servers
  - Extending the metric to measure other system processes such as query processing.

# Q & A

Seon Ho Kim, Ph.D.
seonkim@usc.edu

**Thank you!**