FINE-GRAIN OFFLOADING TO THE CLOUD

Presenter: Shawn Ju
Simply DSM Support Offloading

Mobile App

Cloud Daemon

Distributed Shared Memory

Memory

Object

variable_X

method_Y()

Memory (replica)

Object

variable_X

method_Y()

Continuous Replication

Dynamic + Fine-grain Offloading
Advantages of Dynamic + Fine-Grain

• Dynamic
  • Can offload arbitrary work at runtime
  • Can optimize resource utilization at runtime

• Fine-grain
  • At the level of a method invocation
  • Feels more responsive when failure requires local restart to fix
  • Useful for leveling out utilization while providing low-latency serves for end-users
Challenges

• Latency
  • Dictates granularity. If update takes more time than not offloading, don't offloading.

• Bandwidth
  • Only send deltas, but determine delta encoding has computation cost

• Compute
  • Compression can be computationally expensive.

• Battery
  • Should not end up consuming more battery budget than not offloading
Compressive Sensing

\[ \Phi \]

\[ M \times N \]
Random sampling matrix
\((M < N)\)

\[ x \]
\[ N \times 1 \]
K-sparse signal

\[ = \]

\[ y \]
\[ M \times 1 \]
Compressive samples
Compressive Sensing Decoding

- L2-norm minimization: Fast, Incorrect
- L0-norm minimization: Intractable, Correct
- L1-norm minimization: Tractable, Correct (usually)

A linear program solves the L1-norm minimization:

$$\min_{\hat{x} \in \mathbb{R}^N} \|\hat{x}\|_1 \quad \text{s.t.} \quad y = \Phi \hat{x}$$
Novel Characteristic

• All-in-one
  • Delta encoding + compression

• Delta encoding figured out by server, not device
  • Automatically recovered during decoding
  • Just send compressive samples; no more network costs

• Codec is resource-commensurate
  • Device: low-complexity encoder
  • Server: higher complexity decoder
  • Unlike traditional compressors
Comparison

- **Rsync**
  - Compression is an added-up steps
  - Needs multiple-trip to determine delta encoding

- **Compress snapshot**
  - No round trip overheads
  - A whole memory page is compressed

- **Metrics**
  - Replicate 64kB memory block
  - Latency = encoding + network + decoding
  - Compression ratio
Prototype

- Mobile app links against libupshift
- Objects are allocated from privately managed memory heap
- Replication agent manages replication of this memory
- Offloading is done via method swizzling (Objective-C is late binding)
- Offloadable objects are abstracted out into cross-compiled libraries, so server has class definitions
- Address spaces are fully managed by UpShift so we do address translation
Trade-off

- Encoding time
- Transmission time
- Decoding time