Elastic HTML5: Workload Offloading using Cloud-based Web Workers and Storage for Mobile Devices

Xinwen Zhang, Won Jeon, Simon Gibbs & Anugeetha Kunjithapatham

Computer Science Lab.,
Samsung R&D, San Jose CA

Outline

- Overview
  - Motivation: Cloud Power
  - Background: Cloud Computing & HTML5

- Approach
  - Overview
  - Advantages

- Elastic HTML5
  - Web Workers: Creation, Communication & Storage

- Conclusion
  - Challenges & Future Work
  - Summary
  - Related Work
Outline

• Overview
  - Motivation: Cloud Power
  - Background: Cloud Computing & HTML5

• Approach
  - Overview
  - Advantages

• Elastic HTML5
  - Web Workers: Creation, Communication & Storage

• Conclusion
  - Challenges & Future Work
  - Summary
  - Related Work

Motivation: Cloud Power

The goal of Elastic HTML5 is to enable cross device/cloud app development &
thereby:

• New classes of powerful applications
• New business model
• Transition path to multi/many core
Cloud Computing (1 of 2)

**IT View of Cloud Computing**

cloud = web service platform

- Cloud is a platform for *service delivery*
- Push from services into devices

Cloud Computing (2 of 2)

**Proposed CE View of Cloud Computing**

cloud = data/core/network center + API

- Cloud is a platform for *new applications* that run across the cloud and device (“elastic applications”)
- API exposes cloud to device apps
- *Expand the device into the cloud*
Background: **HTML5**

- W3C’s 5th major revision of web’s core language
  - Important features introduced:

1. **Web workers**
   - Forked from main browsing context in browser runtime
     - Communication via `postMessage()` & `onMessage()`
     - Could be implemented as separate threads or processes

2. **Local storage**
   - HTML5 apps can use local storage on client
     - SQL database
     - App-specific data

- No consideration (yet) for running web workers in the Cloud
Outline

- Overview
  - Motivation: Cloud Power
  - Background: Cloud Computing & HTML5

- Approach
  - Overview
  - Advantages

- Elastic HTML5
  - Web Workers: Creation, Communication & Storage

- Conclusion
  - Challenges & Future Work
  - Summary
  - Related Work

Approach: *Elastic HTML*

- Migrate web worker & storage (for html5 app on device) to the Cloud
- Communicate via existing HTML5 APIs (message-based communication channels)
- Access local database storage with existing APIs, although storage can be at device and/or cloud
- Access external web servers with same security policies as in client side browser
Approach: Advantages

- Transparently augment browser capability with cloud compute and storage
- No new language introduced to application developers
- No change to existing web programming model: HTML/CSS/JavaScript
- Facilitates cross-platform development
- Simple extension / integration with web apps (Web Worker and storage APIs)
- Simple extension / integration with mobile platforms (browser-based)

Outline

- Overview
  - Motivation: Cloud Power
  - Background: Cloud Computing & HTML5

- Approach
  - Overview
  - Advantages

- Elastic HTML5
  - Web Workers: Creation, Communication & Storage

- Conclusion
  - Challenges & Future Work
  - Summary
  - Related Work
Web Workers: Creation

- Device-side proxy manager decides where to launch the web worker
- If it decides to launch the target web worker at cloud, it sends the request to the cloud proxy manager
- Cloud proxy manager can fetch web worker code directly from origin server
- Proxy managers respect browser security, such as SOP

Web Workers: Communication

- Runtime message flow:
  - Proxy managers route & relay messages between web workers & main thread
  - Communication between workers through interfaces specified by HTML5
  - Communication between proxy managers is bidirectional, proprietary
Web Workers: Storage

- App/worker-specific data storage
  - Interfaces specified by HTML5 used for local database storage & access
  - Web worker invokes openDB(), proxy manager checks for DB on both device & cloud
    - If DB doesn’t exist, DB created locally (local to worker) (transactions via DB.transaction())
    - If DB exists, the elasticity layer /proxy managers routes message to corresponding DB & results to corresponding worker
  - DB on cloud works like local storage to device web workers
  - DB replication & synchronization has to done, if required

Outline

- Overview
  - Motivation: Cloud Power
  - Background: Cloud Computing & HTML5

- Approach
  - Overview
  - Advantages

- Elastic HTML5
  - Web Workers: Creation, Communication & Storage

- Conclusion
  - Challenges & Future Work
  - Summary
  - Related Work
Challenges & Future Work

- **Cost Model [19]**
  - Which worker runs on device / cloud
  - What is the worker pattern: pools, shadowing, task splitting, aggregation
  - When to migrate a worker (from/to device/cloud)
  - Minimize power consumption on device
  - Minimize network latency (esp for user interactive apps)

- **Migration, Replication of Data & Code**
  - Database integrity & synchronization
  - Run-time application state migration
  - Transparent & seamless for rich mobile user experience
  - Offline access

- **Security and privacy**
  - Mutual authentication between workers on device and cloud
  - Authorization delegation to workers running on public cloud
    - Lightweight protocol to distribute shared secrets, session keys [20]
  - Trust between workers on the cloud - open problem

Summary

- **Techniques for Elastic Web Workers & Elastic Storage**
  - Bridges resource-constrained CE devices & Cloud
  - Powerful web applications for next-generation mobile devices

- **Several benefits to use HTML5**

- **Several benefits to Mobile + Cloud approach**
  - Seamlessly expand / shrink device capability
  - Users dynamically control device capability (based on battery, cost, latency etc.)
  - Device need not be designed to satisfy high demanding app
  - App Dependability: apps migrated to cloud when low battery/ weak signal
  - Apps Future Proofed: multi-core apps can run on device when ready, lifetime extended
Related Work: Mobile + Cloud Approaches

- CloneCloud (HotCloud’09)
  - Clone of phone image at cloud

- Splitting applications between device and cloud
  - Dynamic partitioning of applications (MCS’10)
  - Dynamic remote method invocation with managed code (Mobisys’10)

- Dynamic Composable Computing (HotMobile’08)
  - Dynamic composition of functions with mobile devices and surrogates.

- Cloudlet (PVC’09)
  - Offloading VM to proximate infrastructure
  - 60-90s on VM synthesis

- HW-supported VM migration (Atom) (MobiCase’09)
  - Focus on mobility of app

- ... 

- Elastic Device/Application
  - On application level
  - Dynamic execution configuration
  - More flexible and easy for parallel...