Cloud Middleware and MBaaS

COSC349—Cloud Computing Architecture

David Eyers

Learning objectives

- Outline the **usefulness of middleware** for developing applications that use cloud computing
- Contrast Apple **CloudKit** and Google **Firebase** in terms of relationships between provider, tenants and clients
- Describe typical **MBaaS services**
  - MBaaS—Mobile Back-end as a Service (seen in XaaS lecture)
- Sketch **pricing approach** of CloudKit versus Firebase

Middleware for cloud computing

- Middleware: OS-like functionality for apps beyond OS
  - e.g., OS accesses local files; **middleware accesses cloud files**
    - (assumption that OS isn't cloud aware is less realistic over time…)
  - Middleware often **eases translation between platforms**
  - Typically use middleware via a software library
    - … contrast this with programming directly against cloud APIs
- Focus on two (Mobile) Backend as a Service offerings:
  - Apple's **CloudKit** (2014)—iCloud launched 2011
  - Google's **Firebase** (~2014)—Firebase launched in 2011

MBaaS versus typical AWS services

- Amazon has a relationship with the tenant
  - … but does not explicitly have any link with tenants' clients
- Contrast MBaaS from Apple, Google, etc.:
  - company has **relationship with the tenant** (*e.g.*, iOS app dev.)
  - company also has a **relationship with the client**
- Differences between Apple, Google, etc.
  - Apple has a **financial link** by selling kit to tenants' clients
  - Google interests in **cross-service linkage**—across search, *etc*.
  - (Amazon doesn't seem to leverage Amazon+AWS client links)

Apple CloudKit

- From XaaS lecture: it's Backend as a Service (BaaS)
- Apple has always maintained **deep vertical integration**
  - Hardware; OS; even programming languages and compilers
    - PL: Objective-C, then Swift; Apple strongly supports LLVM compilers

- Application developer can rely on CloudKit for:
  - **Databases**—Apple provides public and private instance
  - **Authentication**—Apple provides authenticated users
  - **File storage**—Apple allows use of iCloud Drive

FYI: CloudKit code example (Swift)

```
private func fetchItems() {
    let privateDatabase = CKContainer.default().privateCloudDatabase

    // Initialise Query
    let reference = CKReference(recordID: list.recordID, action: .deleteSelf)
    let query = CKQuery(recordType: RecordTypeItems,
                        predicate: NSPredicate(format: "list == %@", reference))

    // Configure Query
    query.sortDescriptors = [NSSortDescriptor(key: "name", ascending: true)]

    // Perform Query
    privateDatabase.perform(query, inZoneWith: nil) { (records, error) -> Void in
        DispatchQueue.main.sync {
            self.processResponseForQuery(records, error: error)
        }
    }
}
```

Common CloudKit database objects

- **CKContainer**—security sandbox for applications
- **CKDatabase**—a key-value store with constraint support
  - private: sensitive user information (see code on previous slide)
  - public: shared data
- **CKReference**—like constraints in relational DB model
- **CKRecord**—key-value pair inside your database
  - **CKRecordZone**—app has zone; can define others
  - **CKRecordIdentifier**—unique label of record
- Note the use of asynchronous data retrieval

CloudKit notifications and CloudKit JS

- CloudKit facilitates **push notifications** to clients
  - (Complete code, such as asking user permission for notifications, omitted)

```
let subscription = CKQuerySubscription(…)
let notificationInfo = CKNotificationInfo()
notificationInfo.alertBody = “Notification received"
notificationInfo.shouldBadge = true
notificationInfo.soundName = "default"
subscription.notificationInfo = notificationInfo

CKContainer.default().publicCloudDatabase.save(subscription,
    completionHandler: { subscription, error in
        if error == nil { /* subscription saved */ }
        else { /* error */ }
    } )
```

  - History: **Apple Push Notification** service first on iOS 3 in 2009

- CloudKit JS: can interact with iCloud using web app.
  - Still need to set up everything in Apple Xcode
  - Get API token and register application

CloudKit backend for databases, etc.

- Not entirely clear what Apple iCloud backend uses
  - *e.g.*, see Cassandra, MongoDB, HBase, and Couchbase job ads
  - Good evidence different products used for different tasks
- Apple's Apache **Cassandra use in 2014**:
  - More than 75,000 servers; >10PB data; >1M ops/sec
  - Timing implies mostly Apple iCloud use, not use by CloudKit
  - (Cassanda project was open-sourced by Facebook in 2008)
- **FoundationDB bought by Apple**; re-open-sourced 2018
  - Compared to Cassandra: supports ACID transactions; in memory

CloudKit pricing

- Free tier: 10GB **assets**; 100MB **DB**; 2GB **transfer**; 40 **req/s**
  - User data and docs not included in app's usage (it's in iCloud)
- Free tier scales based on active users (iOS $ spenders)
  - Per user: 250MB assets; 2.5MB DB; 50MB transfer; 0.0001req/s
    - 500K users? 125TB assets; 1.2TB DB; 25TB transfer; 50 req/s
    - 4M users? 1PB assets; 10TB DB; 200TB transfer; 400 req/s
      - Resource ceiling at 4M users: more users, just get smaller slice
- Overage charges
  - Assets $0.03/GB; DB $3.00/GB; $0.10/GB transfer; $100 10req/s

Google's Firebase MBaaS

- Google Cloud started more 'AWS' than 'CloudKit'
  - Google acquired Firebase in 2014—it had started in 2011;
  - has merged aspects of Firebase back-end into Google Cloud

- Firebase **unifies MBaaS**: iOS, Android, web, Unity, C++…
  - Provides file storage, databases, authentication, messaging, testing, profiling, debugging, and many more
  - (Actually, multiple types of DB: Realtime DB and Firestore…)
    - Firebase launched to support **machine-to-machine Realtime DB**
    - Subsequent analytics have presumably led to redesign…

Google's Firebase has three pricing types

- **Spark**—free, and **Blaze**—pay as you go
  - Blaze includes Spark's free quota (RIP Flame fixed price plan)

	Free	PAYG
Phone auth	10k/mth	$0.06/check
Firestore data	1GiB	$0.18/GiB
Functions invocations	125K/mth	$0.40/million
Realtime DB connections	100	200K/database
Storage (data)	5GB	$0.026/GB
Storage (downloads)	1GB/day	$0.12/GB

Firebase services—application building

- Typical services for building applications
  - **Cloud Firestore**—NoSQL DB
  - **Cloud Functions**—like AWS Lambda
  - **Authentication**—passwords + OAuth
  - **Hosting**—CDN for web content
  - **Cloud Storage**—object storage like Amazon S3
  - **Realtime Database**—original low-latency state synchroniser
  - **Cloud messaging**—push notifications
    - Replaced Google Cloud Messaging for push to Android, with service that does push to iOS, Android and web

Firebase services—QC and analytics

- Quality Control (QC) for applications
  - **Crashlytics**—aggregates crash reporting; realtime notification
  - **Performance monitoring**—network and device tracing
  - **Test lab**—run code on real+virtual devices; CI; auto-testing
- Services linked to analytics
  - **Google Analytics**—like for websites, study your customers
  - **Predictions**—automatically group your users for manipulation
  - **Remote Config**—dynamic customisation: language; specials…
  - **Dynamic Links**—from mobile website to deep link in app

Firebase beta offerings

- **A/B testing**—modify your app and measure effect
  - Allows for incremental development of features
- **In-app messaging**—manipulate users while using app.
  - Messages based on model of user behaviour and interests
- **Firebase ML**—machine learning toolkit
  - Provides for on-device and in-cloud APIs across platforms
  - Supports upload of custom TensorFlow Lite models
    - TensorFlow is another data flow system, for maths-heavy computing
    - TensorFlow Lite uses GPU inference engine running on smartphones