SAP Mobile Platform 3.0 Performance Guidance

Choosing the right mobile technology for your application

Audience

This document assumes that the reader is familiar with OData and MBO development. See the SAP Mobile Platform product documentation for a thorough background.

Purpose

This document provides guidance for choosing an appropriate mobile platform technology, based on a discussion of their performance characteristics. SAP Mobile Platform 3.0 supports two primary methods of building custom applications - OData and Mobile Business Object (MBO).[1] The performance guidance provided here is intended to help developers make an informed decision before adopting one technology over another. Consider technology design and performance characteristics carefully when deciding how to build your application, to provide an appropriate user experience.

The performance data represents a relative comparison of OData and MBO. This analysis was performed based on tests from a 2013 mid-range Android tablet operating on WiFi and a local area network. While devices vary in absolute terms, the relative comparison of technology is the intent of this discussion. To accurately predict your own user experience with a specific device, take measurements early in the development process with representative application data sizes.

OData Formats

The mobile platform continues to pursue the SAP emphasis on open standards, specifically related to OData. SAP NetWeaver Gateway (2.0 SP4 or later) and SAP Mobile Platform 3.0 support two data formats: XML and JSON. An HTTP header provided by the client determines the format retrieved from the server.

```
XML → "Accept", "application/atom+xml"
```

```
JSON → "Accept", "application/json"
```

The JSON data format is designed to be compact, while XML is verbose. The performance gain in using JSON is significant on mobile devices where memory, processing power, and network resources are relatively sparse.

[1] A third method of exchanging mobile data, Agentry, is used in certain bespoke SAP applications but is not considered a primary custom development approach and is therefore not discussed in this context.

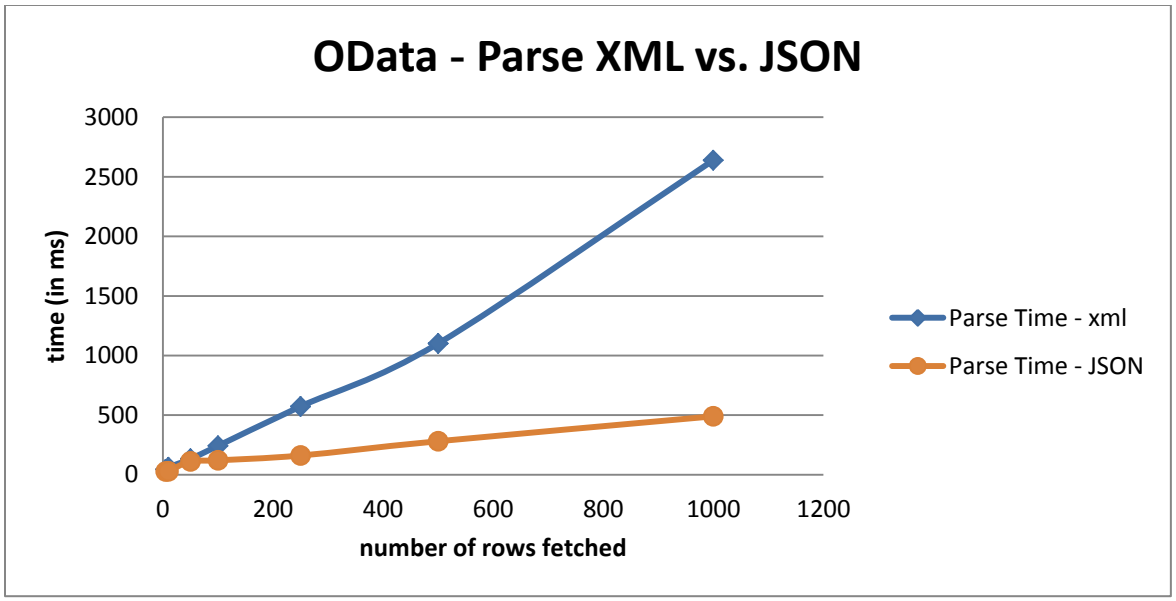

Figure 1 Device Parse Times: OData Formats

The data in the parse chart clearly shows JSON as the preferred format for mobile applications, especially when data grows to more than a few records.

Offline OData

The latest release of SAP Mobile Platform Server and SAP Mobile Platform SDK now support limited OData offline capabilities. You can use these offline facilities to synchronize data queried from a NetWeaver Gateway server without retrieving the full data set to the device with each new request. NetWeaver Gateway facilitates client-side OData synchronization by allowing a developer to implement a “delta query” capable service. Delta-capable Gateway queries supply a token to the client as part of the original results set. The delta token may be returned to the service, to identify and retrieve only the changes made since the token was last retrieved from the server. An SAP Mobile Platform SDK developer may merge record change sets based on the delta query with the original record data stored on the device, maintaining an updated data view as of the time of the last query. The device SDK actions of fetching, parsing, merging, and storing may continue as long as the client’s delta tokens are honored by the server.

MBO Synchronization

Every version of the mobile platform supports a proprietary protocol known as MobiLink. The MobiLink protocol is an efficient record synchronization from a staging database or back end to an SAP UltraLite or SQL Anywhere database on the device. Unlike offline OData, which is passed in JSON or XML text and parsed, MobiLink moves records in a form that is optimized for insertion into the device database. The single MobiLink synchronization step equates to all of the OData developer actions of fetch, parse, merge, and store.

Comparing Offline OData and MBO Performance

To compare the performance of these technologies, we need to consider both download and update phases. The comparison for download synchronization represented in this document used a resident device database size of approximately 1,500 records.

Synchronization Download

The high-level analysis suggests that OData processing using a SQLite database performs well with a relatively small number of records where parsing, merging, and storing activity are kept to a minimum. The optimized MobiLink protocol and the finely tuned pairing of the UltraLite database is efficient when synchronizing large numbers of records, but also has a fixed overhead even for a small number of change records.

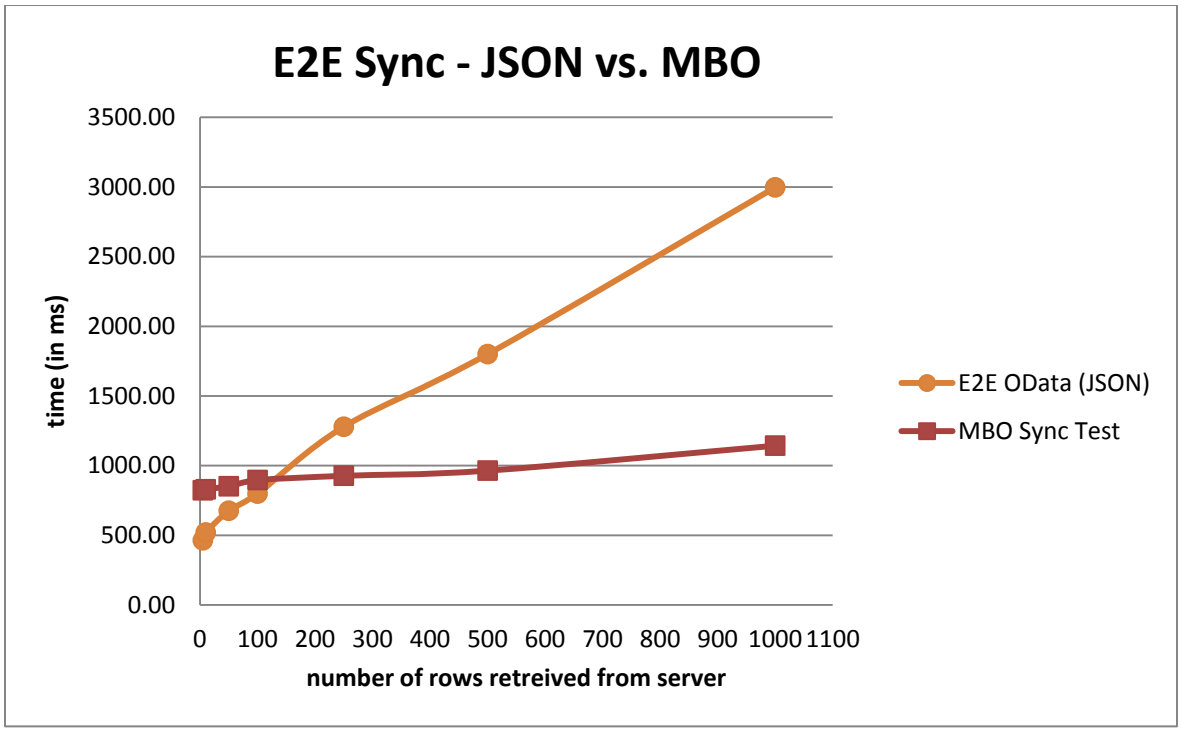

Figure 2 Offline OData vs. MBO

Because the on-device OData operations operate on collections that are entirely executed in memory and stored as an atomic unit, the cost of synchronization (memory and processing) grows linearly as record collections become larger. This effect is true whether the delta changes become larger, or whether the on-device storage grows because OData SDK operations must read the delta changes and device data into memory to perform the merge and then store the result to disk. Another consideration is that due to the present in-memory nature of OData operations in the SDK, it is impossible to read a single detail record of a large record set without reading the entire collection into memory.

Synchronization Updates

MBO supports two updates modes: synchronous and asynchronous (the default). The synchronization mode is configured on the device using the Object API. MBO synchronous updates are sent in batches to the server, but executed against the back end (SAP system, Web service, or database) in a serial time-ordered fashion on a single thread. The MBO synchronous upload is automatically followed by a download and merge of any changes from the server. In contrast, MBO asynchronous updates are shipped to the server and executed in separate threads without requiring the device to wait for the updates to be executed. This is sometimes referred to as a "fire and forget" upload, although a notification is sent to the device once the updates have been executed with a resultant log. Synchronous updates are much more costly and should therefore be avoided unless required by business circumstances.

The OData protocol supports batch update processing from the device to the back end, if the service also supports it. The efficiency of the processing on the back end is implementation dependent. In many cases, batch processing of an OData request can be converted to a single efficient transaction.

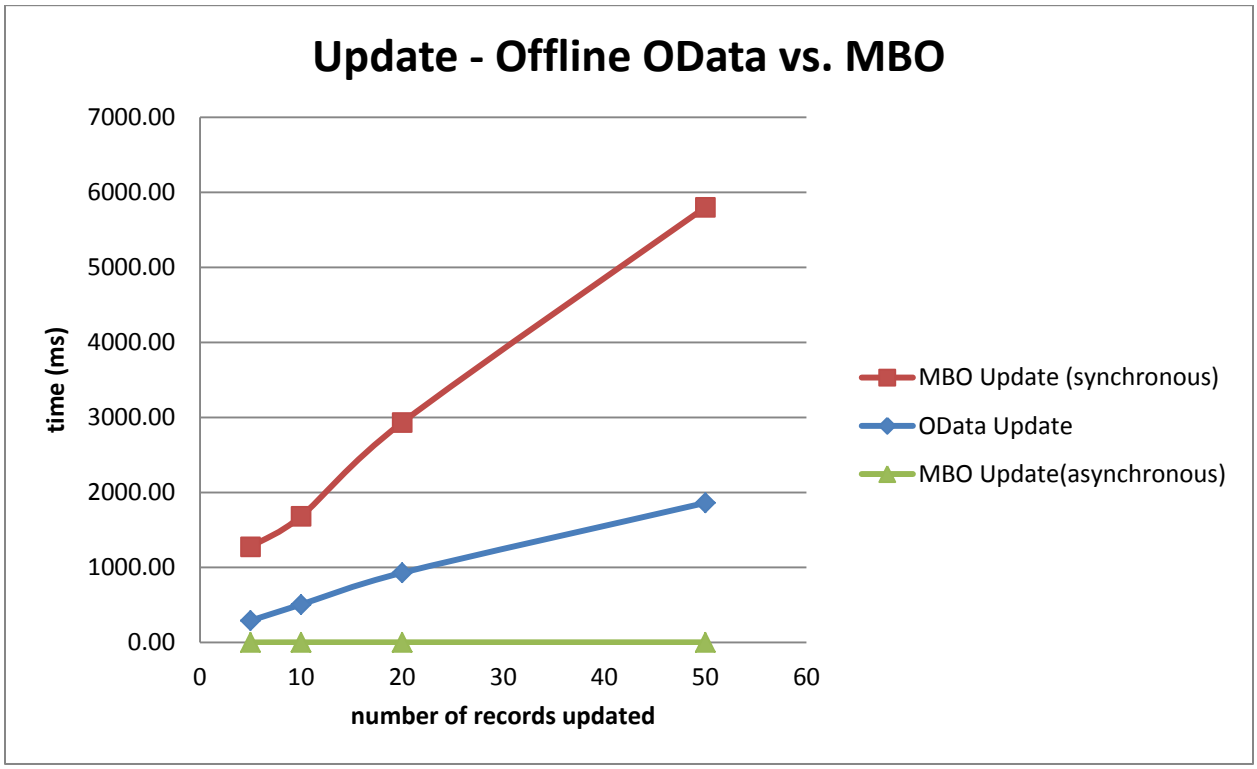

Figure 3 Update OData vs. MBO

In our analysis, we placed up to 50 records on the device and measured the update phase of the various technologies. For comparison purposes, the updates for OData and asynchronous MBO phases do not include download phases. The data suggests that OData batch updates are relatively efficient, especially when compared to a synchronous MBO update where each update must be serialized to the back end, followed by a download. An OData update without a subsequent fetch/merge operation, or an asynchronous MBO update provides the best throughput. Depending on the business case, a

subsequent download phase might normally follow an update. To obtain a true measure of the cost of application synchronization, see to the earlier discussion of synchronization download, and add this cost to the update.

Summary

OData is appropriate for applications with relatively small demands on device parsing and merging, either because of small delta changes or device record sets that need to be merged. OData updates should use batch processing where possible.

SAP strongly recommends that developers use a MobiLink-based synchronization when device record sets or delta changes become large *and* when merge operations are required.[2]

[2] If you plan to use MBOs, remember, as noted in the MBO documentation, that data change notifications (DCNs) are by far the most efficient way to update MBOs based on back-end changes. Avoid on-demand fetches or scheduled refreshes into the MBO middleware, as they result in poor performance and scaling.

www.sap.com