
Google™
09
I/O

Evolution of the Google Data Protocol

Sven Mawson
05/27/2009

Building More Efficient Applications

- **Introduction**
- Demo: Photo Shuffle
- Introducing JSONC
- Introducing Partial GET
- ETags and Google Data APIs
- Coming Soon
- Q/A

The Google Data APIs

- **Based on common standards**

The Google Data APIs

- Based on common standards
  - **AtomPub**

The Google Data APIs

- Based on common standards
  - AtomPub
  - **RSS**

The Google Data APIs

- Based on common standards
  - AtomPub
  - RSS
  - **JSON**

The Google Data APIs

- Based on common standards
- **Includes many Google Services**

The Google Data APIs

- Based on common standards
- Includes many Google Services
  - **YouTube**

The Google Data APIs

- Based on common standards
- Includes many Google Services
  - YouTube
  - **Picasa Web Albums**

The Google Data APIs

- Based on common standards
- Includes many Google Services
  - YouTube
  - Picasa Web Albums
  - **Calendar**

The Google Data APIs

- Based on common standards
- Includes many Google Services
  - YouTube
  - Picasa Web Albums
  - Calendar
  - **Docs**

The Google Data APIs

- Based on common standards
- Includes many Google Services
  - YouTube
  - Picasa Web Albums
  - Calendar
  - Docs
  - ...

The Google Data APIs

- Based on common standards
- Includes many Google Services
- **http://code.google.com/apis/gdata/**

Building More Efficient Applications

- **What kinds of applications need efficiency?**

Building More Efficient Applications

- What kind of applications need efficiency?
- **How can we make them more efficient?**

Building More Efficient Applications

- What kind of applications need efficiency?
- How can we make them more efficient?
  - **Reduce Number of Requests**

Building More Efficient Applications

- What kind of applications need efficiency?
- How can we make them more efficient?
  - Reduce Number of Requests
  - **Reduce Bandwidth**

Building More Efficient Applications

- What kind of applications need efficiency?
- How can we make them more efficient?
  - Reduce Number of Requests
  - Reduce Bandwidth
  - **Reduce Latency**

OK, let's see an example

Building More Efficient Applications

- Introduction
- **Demo: Photo Shuffle**
- Introducing JSONC
- Introducing Partial GET
- ETags and Google Data APIs
- Coming Soon
- Q/A

http://photoshuffle.appspot.com

Do the Photo Shuffle!

Background: XML -> JSON -> XML

XML -> JSON Conversion (and back!)

XML:

```
<entry>
 ...
 <media:group>
  <media:content
   url='.../foo.jpg'
   type='image/jpeg'
   medium='image'/>
 <media:credit>
Sven Mawson</media:credit>
  <media:description
   type='plain'>
Cool Photo!</media:
description>
 </media:group>
</entry>
```

JSON:

```
"entry": {
 ...
 "media$group": {
  "media$content": [{
   "url": ".../foo.jpg",
   "type": "image/jpeg",
   "medium": "image"}],
  "media$credit": {
   "$t": "Sven Mawson"},
  "media$description": {
   "type": "plain",
   "$t": "Cool Photo!"}
 }
}
```

XML -> JSON -> XML

- **Elements as JSON objects**

Element -> Object Conversion

XML:

```
<entry>
 ...
 <media:group>
  <media:content
   url='.../foo.jpg'
   type='image/jpeg'
   medium='image'/>
  <media:credit>
Sven Mawson</media:credit>
  <media:description
   type='plain'>
Cool Photo!</media:
description>
 </media:group>
</entry>
```

JSON:

```
"entry": {
 ...
 "media$group": {
  "media$content": [{
   "url": ".../foo.jpg",
   "type": "image/jpeg",
   "medium": "image"}],
  "media$credit": {
   "$t": "Sven Mawson"},
  "media$description": {
   "type": "plain",
   "$t": "Cool Photo!"}
 }
}
```

XML -> JSON -> XML

- Elements as JSON objects
- **Attributes as JSON properties**

Attribute -> Property Conversion

XML:

```
<entry>
 ...
 <media:group>
  <media:content
   url='.../foo.jpg'
   type='image/jpeg'
   medium='image'/>
 <media:credit>
Sven Mawson</media:credit>
  <media:description
   type='plain'>
Cool Photo!</media:
description>
 </media:group>
</entry>
```

JSON:

```
"entry": {
 ...
 "media$group": {
  "media$content": [{
   "url": ".../foo.jpg",
    "type": "image/jpeg",
    "medium": "image"}],
  "media$credit": {
    "$t": "Sven Mawson"},
  "media$description": {
    "type": "plain",
    "$t": "Cool Photo!"}
 }
}
```

XML -> JSON -> XML

- Elements as JSON objects
- Attributes as JSON properties
- **Text content as $t**

Text Content -> $t Conversion

XML:

```
<entry>
 ...
 <media:group>
  <media:content
   url='.../foo.jpg'
   type='image/jpeg'
   medium='image'/>
 <media:credit>
Sven Mawson</media:credit>
  <media:description
   type='plain'>
Cool Photo!</media:
description>
 </media:group>
</entry>
```

JSON:

```
"entry": {
 ...
 "media$group": {
  "media$content": [{
   "url": ".../foo.jpg",
   "type": "image/jpeg",
   "medium": "image"}],
  "media$credit": {
   "$t": "Sven Mawson"},
  "media$description": {
   "type": "plain",
   "$t": "Cool Photo!"}
 }
}
```

Photo Shuffle: Using JSON

Loading data using a <script> tag

```
var head =
  document.getElementsByTagName('head')[0];
var script =
  document.createElement('script');
script.src = 'http://picasaweb.google.com'
  + '/data/feed/api/user/' + username
  + '?alt=json&callback=callback';

head.appendChild(script);
```

Receiving the Results

```
function callback(data) {
  var albums = data.feed.entry;

  clearAlbums();
  for (albumKey in albums) {
    addAlbum(albums[albumKey]);
  }
  renderAlbums();
}
```

Parsing the Albums

```
function addAlbum(entry) {
  var album = {};
  album.id = entry.gphoto$id.$t;
  album.feed = entry.link[0].href;
  album.thumb = entry.media$group.
media$thumbnail[0].url;
  album.title = entry.title.$t;
  albums.push(album);
}
```

How can we make this better?

Problems

- **Too many objects**

Objects Everywhere!

```
"media$group": {
 "media$content": [{
  "url": "http://lh3.ggpht.com/foo.jpg",
  "type": "image/jpeg",
  "medium": "image"}],
 "media$credit": {
  "$t": "Me"},
 "media$description": {
  "type": "plain",
  "$t": "Cool!"}
}
```

Problems

- Too many objects
- **Verbose programming model**

JSON Ugliness

```
album.id = entry.gphoto$id.$t;

album.img =
 entry.media$group.media$thumbnail[0].
url;
```

Problems

- Too many objects
- Verbose programming model
- **Too much extra junk sent on each request**

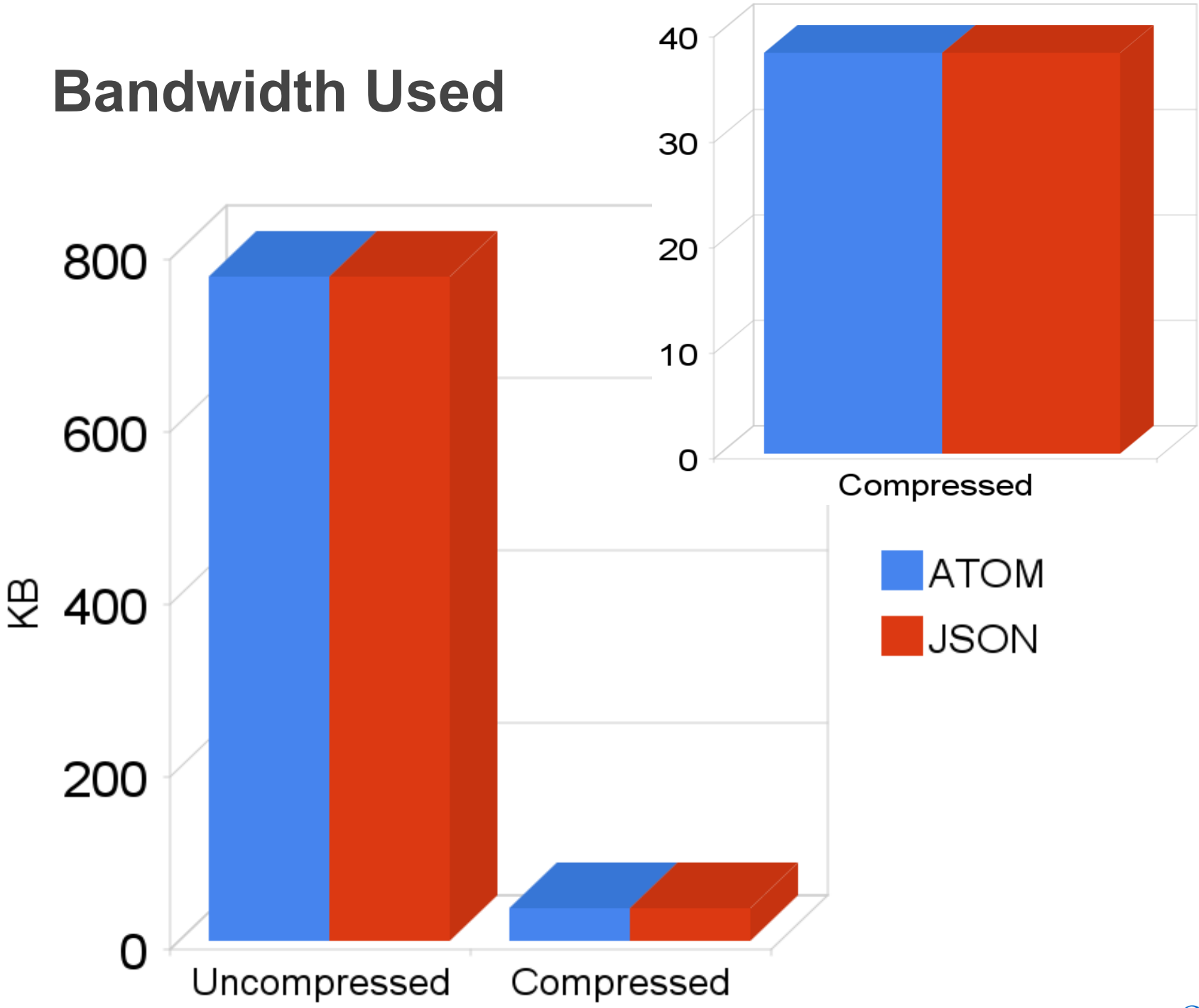
Is this JSON or XML?

```
{
 "version": "1.0",
 "encoding": "UTF-8",
 "entry": {
  "xmlns": "http://www.w3.org/2005/Atom",
  "xmlns$media": "http://search.yahoo.
com/mrss/",
  "xmlns$openSearch": "http://a9.com/-
/spec/opensearchrss/1.0/",
  ...
```

Problems

- Too many objects
- Verbose programming model
- Too much extra junk sent on each request
- **Too much space on the wire**

Bandwidth Used



We Can Do Better!

Building More Efficient Applications

- Introduction
- Demo: Photo Shuffle
- **Introducing JSONC**
- Introducing Partial GET
- ETags and Google Data APIs
- Coming Soon
- Q/A

JSONC: Clean, Compact, and Customizable

- **Minimize the number of JavaScript objects**

JSONC: Clean, Compact, and Customizable

- Minimize the number of JavaScript objects
  - **Simple Single Elements**

Simple Single Elements

JSON:

```
"media$credit": {
  "$t": "Sven Mawson"
}
```

JSONC:

```
"credit":
   "Sven Mawson"
```

JSONC: Clean, Compact, and Customizable

- Minimize the number of JavaScript objects
  - Simple Single Elements
  - **Complex Single Elements**

Complex Single Elements

JSON:

```
"summary": {
 "$t":"Nice!",
 "type": "text"
}
```

JSONC:

```
"summary": "Nice!",
"summaryType": "text"
```

JSONC: Clean, Compact, and Customizable

- Minimize the number of JavaScript objects
  - Simple Single Elements
  - Complex Single Elements
  - **Simple Repeating Elements**

Simple Repeating Elements

JSON:

```
"gphoto$keywords": [
  {"$t": "Google"},
  {"$t": "I/O"}
]
```

JSONC:

```
"keywords": [
  "Google",
  "I/O"
]
```

JSONC: Clean, Compact, and Customizable

- Minimize the number of JavaScript objects
  - Simple Single Elements
  - Complex Single Elements
  - Simple Repeating Elements
  - **Complex Repeating Elements**

Complex Repeating Elements

JSON:

```
"link": [{
 "rel": "edit",
 "type":
"application/atom+xml"
,
 "href": "...
/feeds/myfeed/myentry"

},...]
```

JSONC:

```
"links": {
 "edit": "...
/feeds/myfeed/myentry",
 ...
}
```

Full Conversion of media:group

JSON:

```
"entry": {
 ...
 "media$group": {
  "media$content": [{
   "url": ".../foo.jpg",
   "type": "image/jpeg",
   "medium": "image"}],
  "media$credit": {
   "$t": "Sven Mawson"},
  "media$description": {
   "type": "plain",
   "$t": "Cool!"}
 }
}
```

JSONC:

```
"entry": {
 ...
"media": {
  "content": ".../foo.jpg",
  "contentType":
"image/jpeg",
  "contentMedium": "image",
  "credit": "Sven Mawson",
  "description": "Cool!",
  "descriptionType":
"plain"
 }
}
```

JSONC: Clean, Compact, and Customizable

- Minimize the number of JavaScript objects
  - Simple Single Elements
  - Complex Single Elements
  - Simple Repeating Elements
  - Complex Repeating Elements
- **Remove extra data**

Useless Data: Gone!

Before:

```
"media": {
 "content": ".../foo.jpg",
 "contentType":"image/jpeg",
 "contentMedium": "image",
 "credit": "Sven Mawson",
 "description": "Cool!",
 "descriptionType": "plain"
}
```

After:

```
"media": {
 "content": ".../foo.jpg",
 "contentType":"image/jpeg",
 "credit": "Sven Mawson",
 "description": "Cool!"
}
```

Photo Shuffle: Switching to JSONC

Asking for JSONC

```
script.src = 'http://picasaweb.google.com'
  + '/data/feed/api/user/' + username
  + '?alt=jsonc&callback=callback';
```

Receiving the Results

JSON:

```
function callback(data) {
var albums =
  data.feed.entry;

 clearAlbums();
 for (key in albums) {
  addAlbum(albums[key]);
 }
 renderAlbums();
}
```

JSONC:

```
function callback(data) {
 var albums =
   data.albums;

  clearAlbums();
  for (key in albums) {
    addAlbum(albums[key]);
  }
  renderAlbums();
}
```

Parsing the Albums

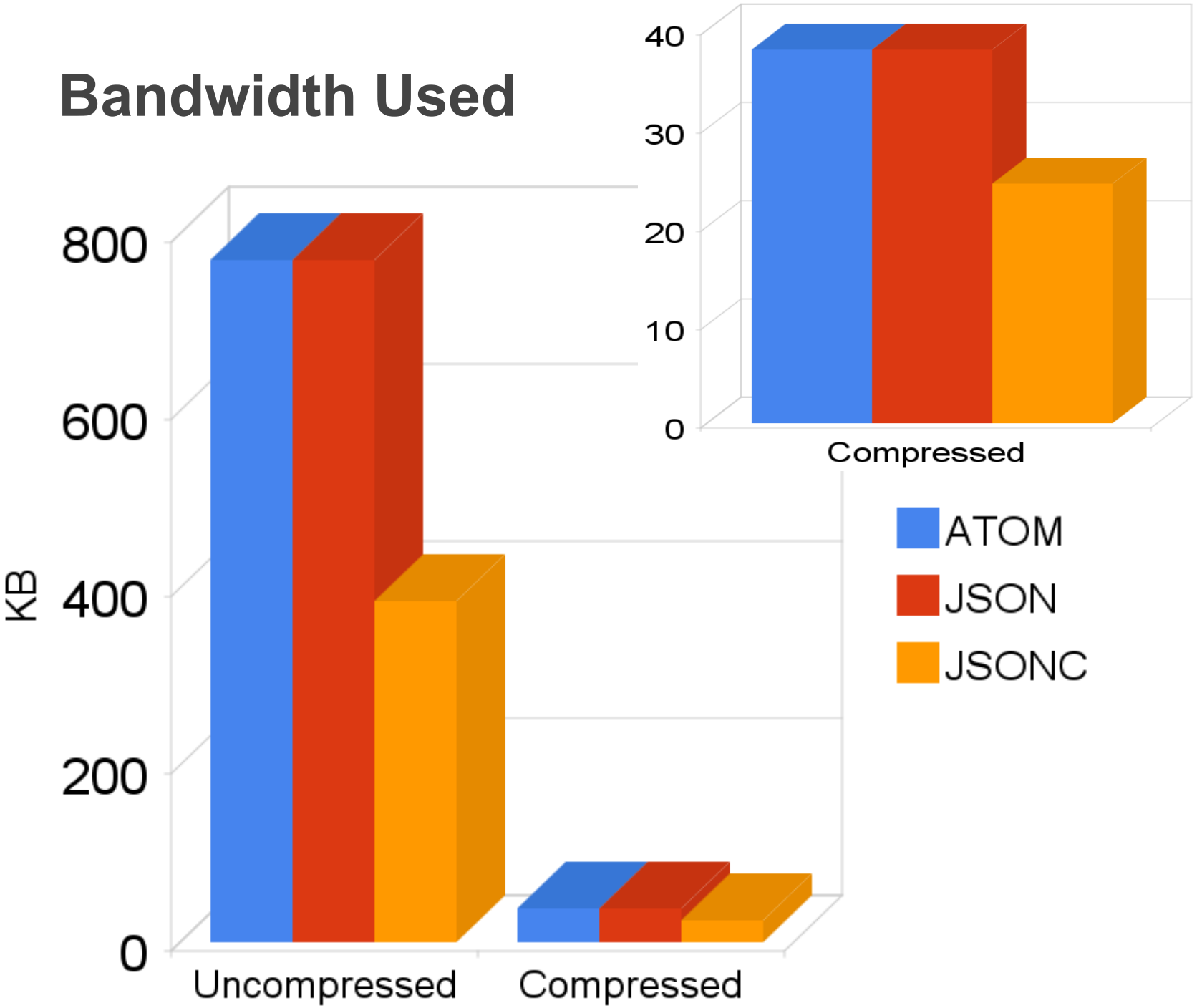
JSON:

```
function addAlbum(entry) {
 var album = {};
 album.id =
   entry.gphoto$id.$t;
 album.feed =
   entry.link[0].href;
 album.thumb =
   entry.media$group
.media$thumbnail[0].url;
 album.title =
   entry.title.$t;
 albums.push(album);
}
```

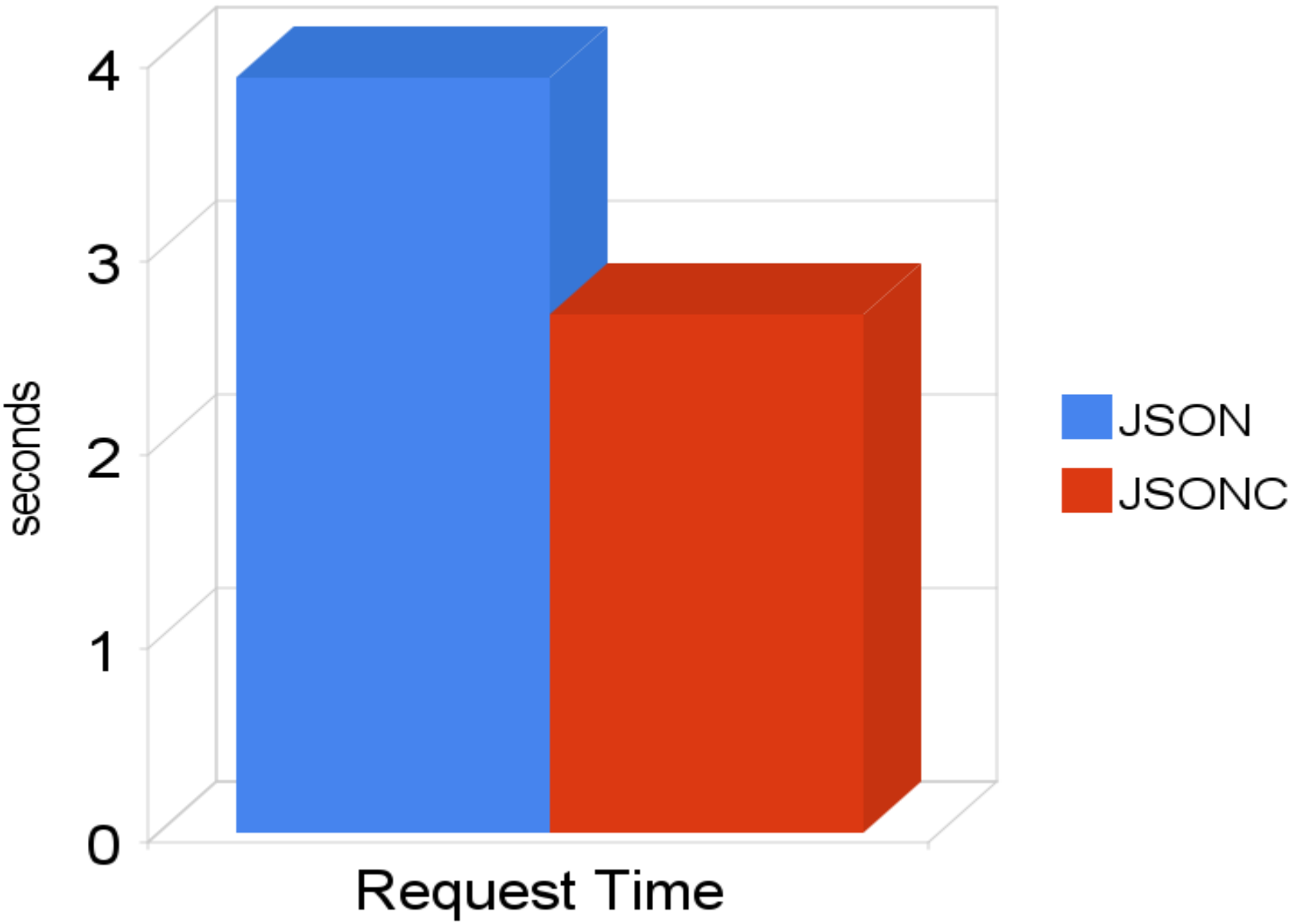
JSONC:

```
function addAlbum(entry) {
 var album = {};
 album.id =
   entry.id;
 album.feed =
   entry.links.feed;
 album.thumb =
   entry.media.thumbnails[0];

 album.title =
   entry.title;
 albums.push(album);
}
```

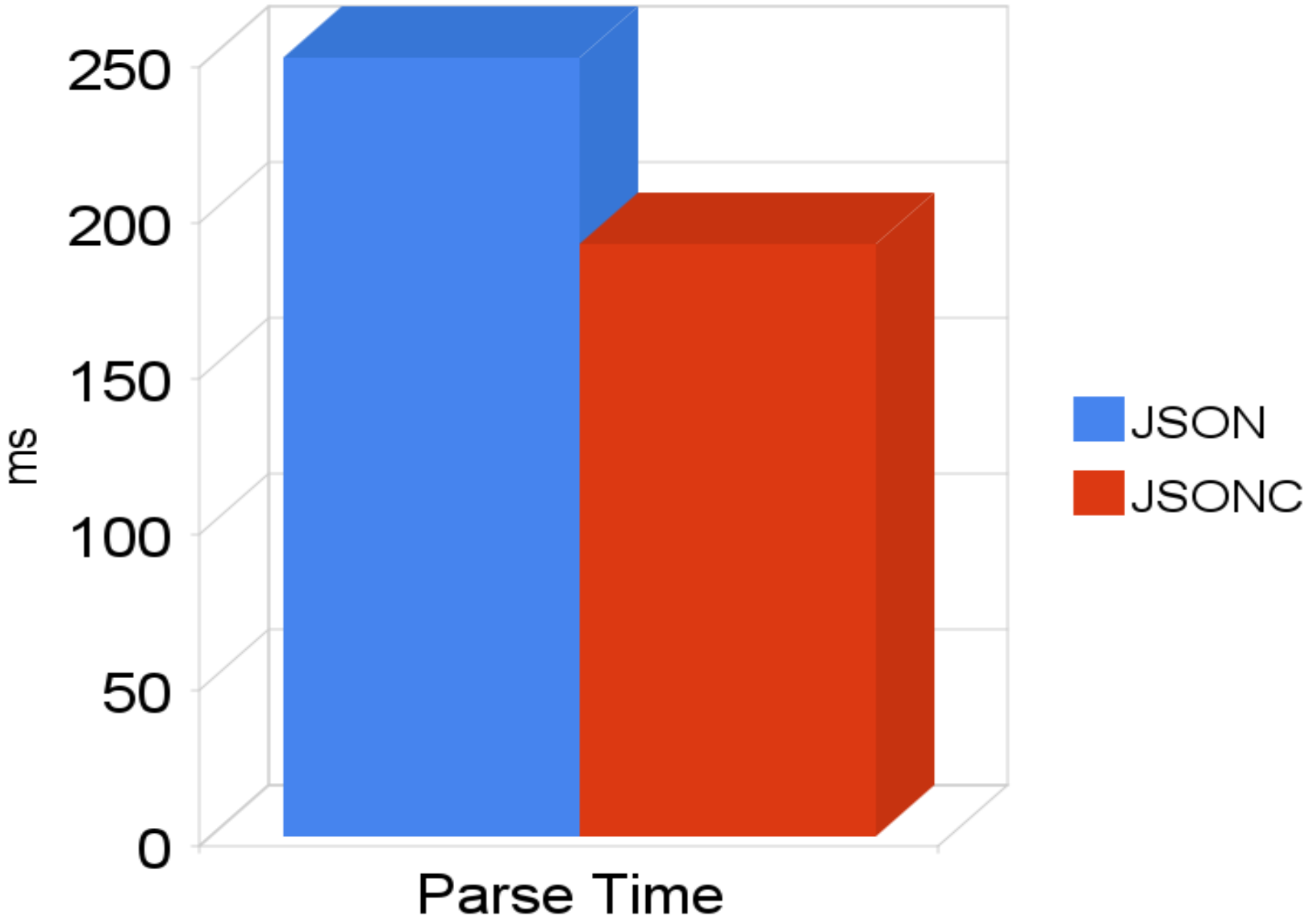
Bandwidth Used



Request Time



Parse Time



Building More Efficient Applications

- Introduction
- Demo: Photo Shuffle
- Introducing JSONC
- **Introducing Partial GET**
- ETags and Google Data APIs
- Coming Soon
- Q/A

Background: RESTful Updates

Read -> Modify -> Write

- GET the entry you'd like to update

GET an entry for update

```
GET /feeds/myfeed/myentry

200 OK
<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'
...>
 <id>..../feeds/myfeed/myentry</id>
 <link rel='edit'
type='application/atom+xml'
   href='.../feeds/myfeed/myentry/1992'/>
 <title>My Photo</title>
 ...other fields...
</entry>
```

Read -> Modify -> Write

- GET the entry you'd like to update
- **Modify the field(s) you'd like to change**

Modify the entry

```
<entry xmlns='http://www.w3.org/2005/Atom'
...>
 <id>..../feeds/myfeed/myentry</id>
 <link rel='edit'
type='application/atom+xml'
       href='...
/feeds/myfeed/myentry/1992'/>
 <title>Favorite Picture</title>
 ...other fields...
</entry>
```

Read -> Modify -> Write

- GET the entry you'd like to update
- Modify the field(s) you'd like to change
- **PUT the entire entry back to the server**

PUT back the updated entry

```
PUT /feeds/myfeed/myentry/1992

<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'
...>
 <id>..../feeds/myfeed/myentry</id>
 <link rel='edit'
type='application/atom+xml'
   href='.../feeds/myfeed/myentry/1992'/>
 <title>Favorite Picture</title>
 ...other fields...
</entry>
```

Read -> Modify -> Write

- GET the entry you'd like to update
- Modify the field(s) you'd like to change
- PUT the entire entry back to the server
- **Repeat if a conflict was detected**

Handling Conflicts

409 Conflict

```
<?xml version='1.0' encoding='UTF-8'?>
<entry xmlns='http://www.w3.org/2005/Atom'
...>
 <id>..../feeds/myfeed/myentry</id>
 <link rel='edit'
type='application/atom+xml'
href='.../feeds/myfeed/myentry/1994'/>
 <title>My Photo</title>
 <summary>Nice Picture!</summary>
 ...other fields...
</entry>
```

Introducing Partial GET

“You can't always get what you want.
But if you try sometimes you might find
You get what you need”

The Rolling Stones

Partial GET

- **Allows clients to specify the elements they want**

The "fields" query parameter

```
GET /feed?fields=id,title,entry
GET /feed/entry?fields=id,title,
media:group,georss:where
```

Partial GET

- Allows clients to specify the elements they want
- **Can specify sub elements**

Specifying subelements

```
GET /feed?fields=entry(media:group(
media:thumbnail))

GET /feed/entry?fields=id,author(name),
georss:where(gml:Pos)
```

Partial GET

- Allows clients to specify the elements they want
- Can specify sub elements
- **Supports conditional filters**

Specifying conditional filters

```
GET /feed?fields=id,entry(media:group(
media:thumbnail[@height=144]))

GET /feed?fields=id,entry
[author/name/text()="Sam"]
```

Partial GET

- Allows clients to specify the elements they want
- Can specify sub elements
- Supports conditional filters
  - **Filters are not queries!**

Sample Response

```
200 OK
Content-Type: application/xml
Etag: {resource-etag}

<gd:partial xmlns:'...' fields='...'>
  <feed gd:etag='...'>
    <title>...</title>
    <entry gd:etag='...'>
      <id>...</id>
      <updated>...</updated>
    </entry>
  </feed>
</gd:partial>
```

Photo Shuffle: Using Partial GET

Asking for a partial response

```
script.src = 'http://picasaweb.google.com'
  + '/data/feed/api/user/' + username
  + '?alt=jsonc&callback=callback'
  + '&fields='
  + 'albums(id,title,links(feed),'
  + 'media(thumbnails))';
```

Receiving the Results

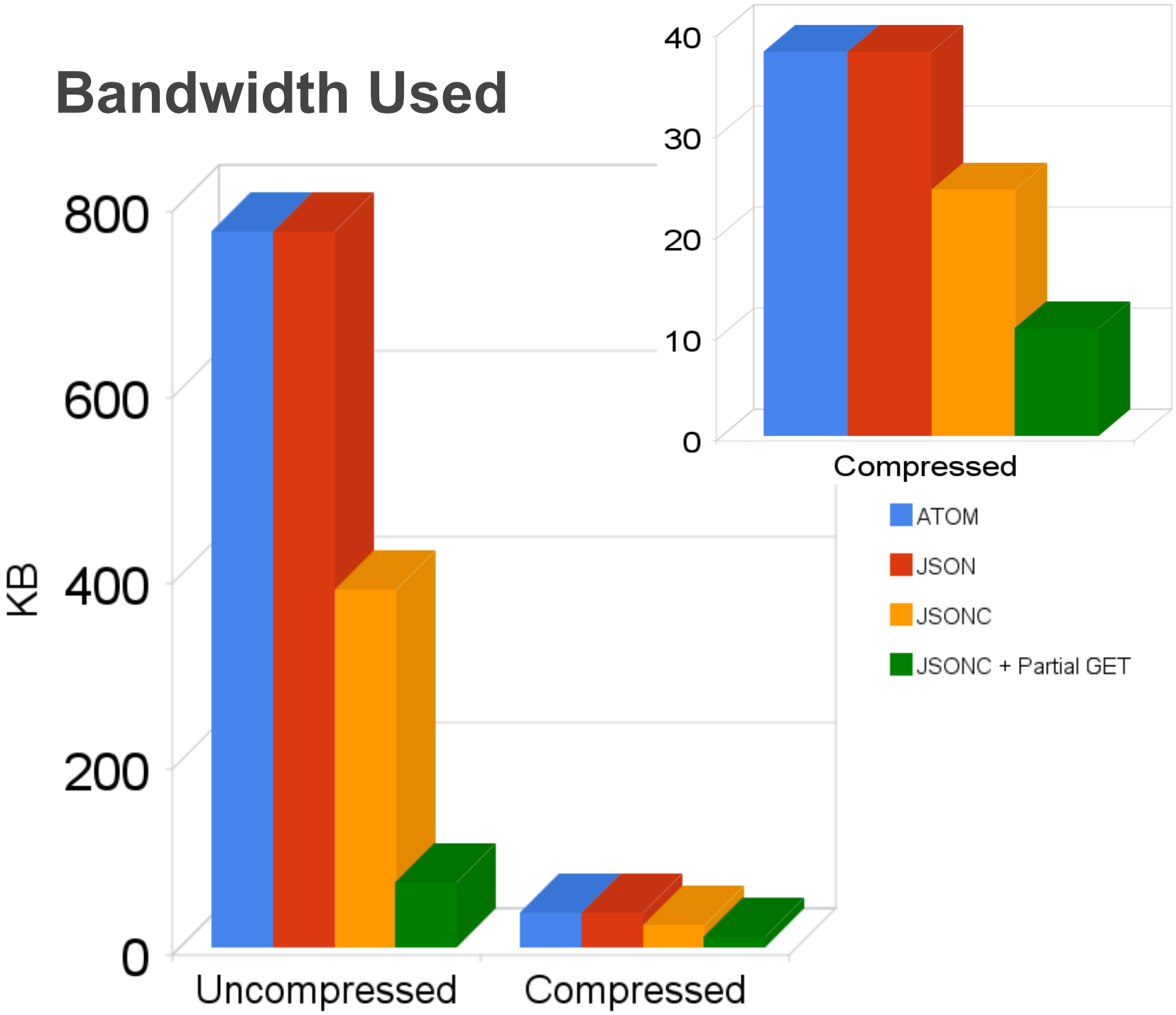
JSONC:

```
function callback(data) {
var albums =
  data.albums;

 clearAlbums();
 for (key in albums) {
  addAlbum(albums[key]);
 }
 renderAlbums();
}
```

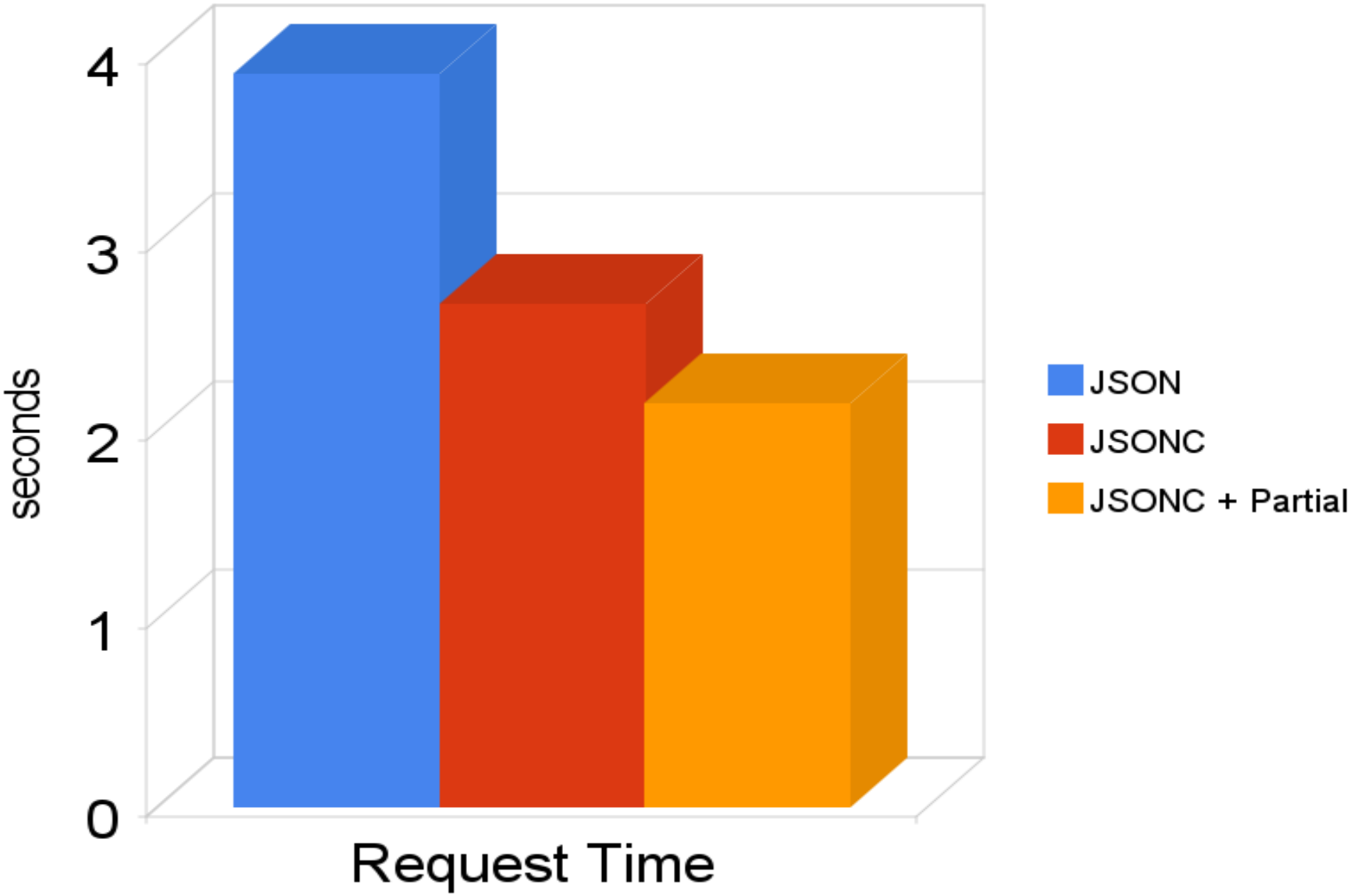
JSONC + Partial:

```
function callback(data) {
 var albums =
   data.feed.albums;

  clearAlbums();
  for (key in albums) {
    addAlbum(albums[key]);
  }
  renderAlbums();
}
```

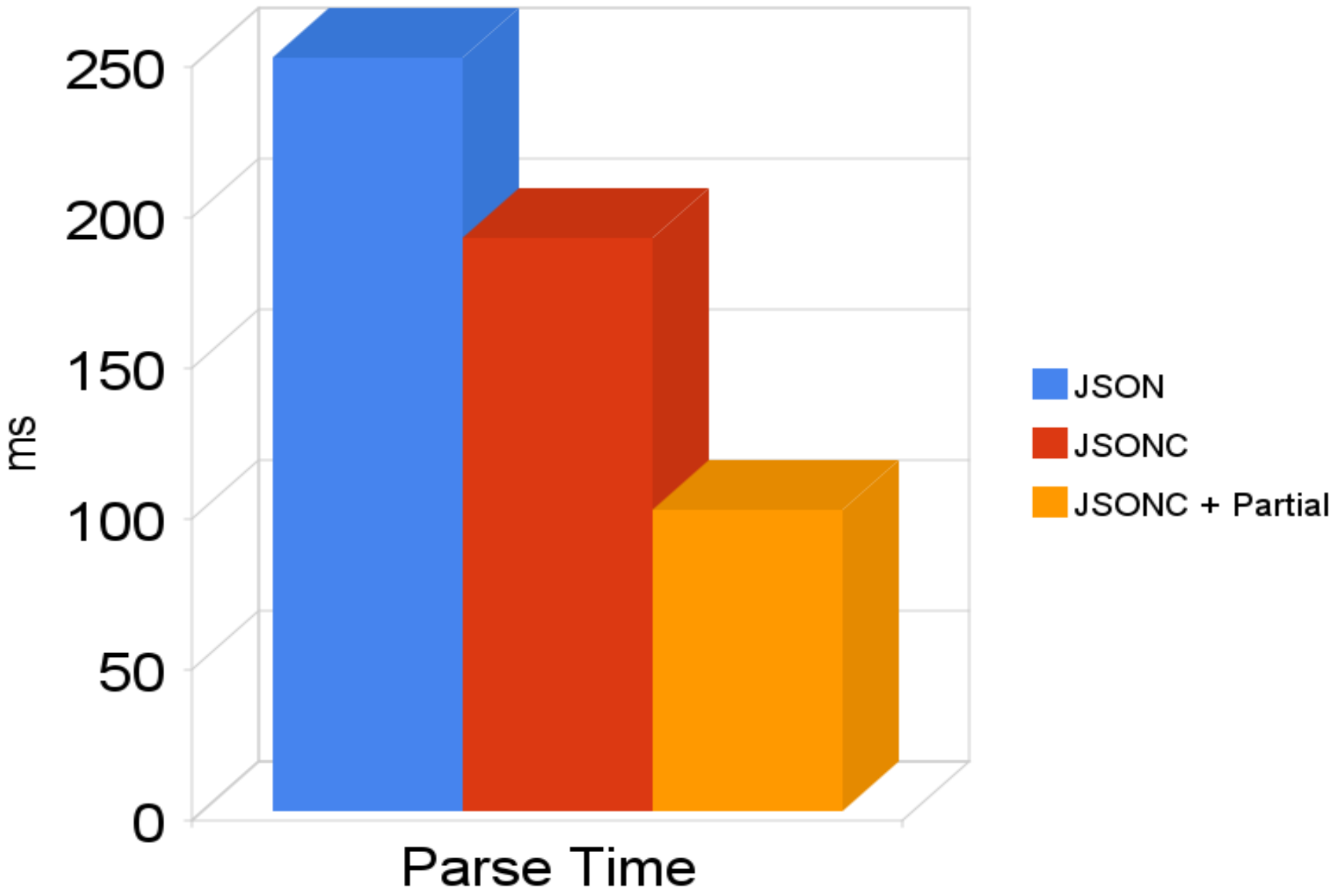
Bandwidth Used



Request Time



Parse Time



Building More Efficient Applications

- Introduction
- Demo: Photo Shuffle
- Introducing JSONC
- Introducing Partial GET
- **ETags and Google Data APIs**
- Coming Soon
- Q/A

Background: Conditional GET

Conditional GET using If-Modified-Since

- **Record Last-Modified header from response**

Recording Last-Modified

```
200 OK
Date: Wed, 13 May 2009 22:48:42 GMT
Last-Modified: Wed, 01 Sep 2004 13:24:52
GMT
...
```

Conditional GET using If-Modified-Since

- Record Last-Modified header from response
- **Send If-Modified-Since header on request**

Requesting If-Modified-Since

```
GET /feeds/myfeed/...

Keep-Alive: 300
...
If-Modified-Since: Wed, 01 Sep 2004 13:
24:52 GMT
...
```

Conditional GET using If-Modified-Since

- Record Last-Modified header from response
- Send If-Modified-Since header on request
- **Use local cache if resource has not changed**

Not Modified Response

```
304 Not Modified

Date: Wed, 13 May 2009 22:52:55 GMT
Expires: Thu, 14 May 2009 04:52:55 GMT
…
```

Problems with If-Modified-Since

- 1 Second granularity

Problems with If-Modified-Since

- 1 Second granularity
- **Server only valid source of time**

Introducing ETags

ETags

- An opaque quoted string

ETags

- An opaque quoted string
  - `ETag: "xyzzy"`

ETags

- An opaque quoted string
- **May be strong or weak**

ETags

- An opaque quoted string
- May be strong or weak
  - `ETag: w/"xyzzy"`

ETags

- An opaque quoted string
- May be strong or weak
- **Returned in the `Etag` header and feed/entry elements**

ETag Response

```
200 OK
Content-Type: application/xml
Etag: "xysddf7893sadf"

<feed xmlns:'...'
      gd:etag="xysddf7893sadf">
  ...
  <entry gd:etag="3jpa246ks88dfx">
    ...
  </entry>
</feed>
```

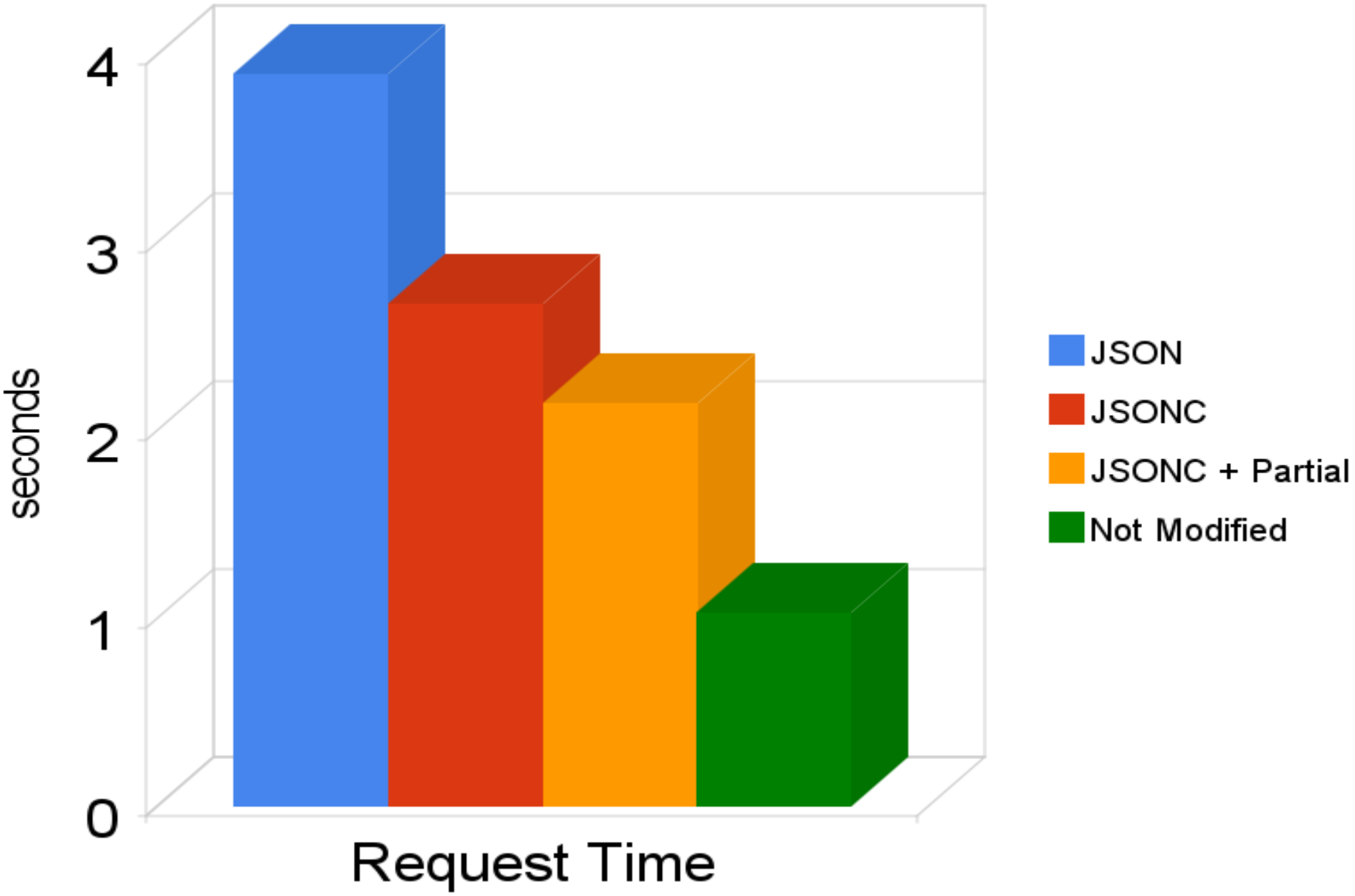
ETags

- An opaque quoted string
- May be strong or weak
- Returned in the `Etag` header and feed/entry elements
- **Use If-None-Match for client-side caching**

If-None-Match Request and Response

```
GET /feeds/feed?v=2 HTTP/1.1
Host: ...
If-None-Match: W/"CUEAQ342szeCp8"
...

HTTP/1.x 304 Not Modified
...
```

Request Time



ETags

- An opaque quoted string
- May be strong or weak
- Returned in the `Etag` header and feed/entry elements
- Use If-None-Match for client-side caching
- **Use If-Match for optimistic concurrency**

If-Match Request and Response

```
PUT /feeds/feed/10233?v=2 HTTP/1.1
Host: ...
If-Match: "SD772sdlk882dkj9"
...

HTTP/1.x 200 OK
...
```

If-Match Failure

```
PUT /feeds/feed/10233?v=2 HTTP/1.1
Host: ...
If-Match: "SD772sdlk882dkj9"
...

HTTP/1.x 412 Precondition Failed
...
```

ETags

- An opaque quoted string
- May be strong or weak
- Returned in the `Etag` header and feed/entry elements
- Use If-None-Match for client-side caching
- Use If-Match for optimistic concurrency
  - **Can use If-Match: * to force update**

Photo Shuffle: Using ETags

Asking for ETags

```
script.src = 'http://picasaweb.google.com'
  + '/data/feed/api/user/' + username
  + '?alt=jsonc&callback=callback'
  + '&fields='
  + 'albums(id,title,links(feed),'
  + 'media(thumbnails))'
  + '&v=2';
```

Building More Efficient Applications

- Introduction
- Demo: Photo Shuffle
- Introducing JSONC
- Introducing Partial GET
- ETags and Google Data APIs
- **Coming Soon**
- Q/A

Coming Soon to an API near you!

Writable JSONC

- **JSONC is not read only!**

Writable JSONC

- JSONC is not read only!
- **No need to translate back to XML for update**

Writable JSONC

- JSONC is not read only!
- No need to translate back to XML for update
- **Supports all write operations**

Writable JSONC

- JSONC is not read only!
- No need to translate back to XML for update
- Supports all write operations
- **Can take advantage of browser capabilities**

Writable JSONC

- JSONC is not read only!
- No need to translate back to XML for update
- Supports all write operations
- Can take advantage of browser capabilities
  - `JSON.stringify(data)`

JSONC Client Library

- **Much smaller than current library**

JSONC Client Library

- Much smaller than current library
- **Authentication support**

JSONC Client Library

- Much smaller than current library
- Authentication support
- **Cross Domain support**

Partial PATCH

- **Partial is not just for GET!**

Partial PATCH

- Partial is not just for GET!
- **Update a partial document using PATCH**

Partial PATCH

- Partial is not just for GET!
- Update a partial document using PATCH
- **Safe `If-Match: *`**

Partial PATCH

- Partial is not just for GET!
- Update a partial document using PATCH
- Safe `If-Match: *`
- **Lower client memory requirements**

Q/A


Google™
09
I/O