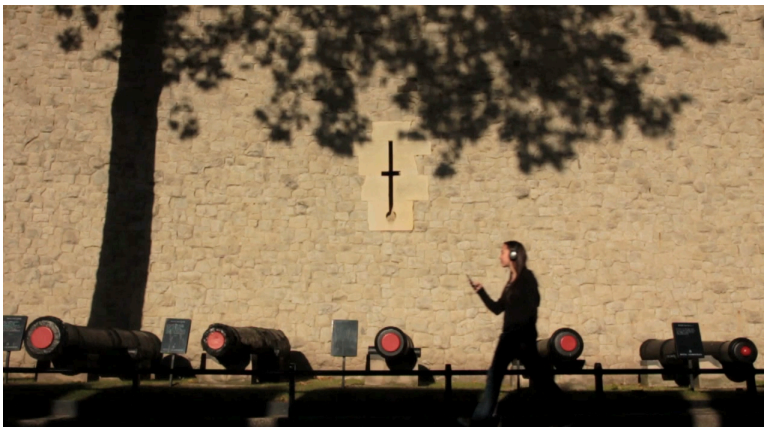
Making scalable location-aware mobile apps

Richard Hull - Managing Director, Calvium

One great use of the mobile phone's ability to know its location in the world is to deliver a rich user experience that relates to that location. For example, a location-aware mobile app can tell stories to the user that convey the rich history and culture of a neighbourhood, as in the Guardian's Streetstories, or influence the moves a player can make in a fun-filled game like Escape from the Tower.



Both of these apps were made with Calvium's AppFurnace, a set of online tools that greatly simplifies the creation of location-aware mobile apps for iPhone and Android. As we work with customers, though, we have become aware that some developers want the experience delivered by their mobile app to go beyond a single neighbourhood or attraction. They want to include lots of geo-triggered content laid out on a national - or even a global - scale. They want their app to deliver value to the user even when it is not running, and they don't want to stop when the user goes indoors.

In this white paper, we will explore the issues involved in scaling content-rich, location-aware mobile apps to meet those emerging needs. To get a feel for the type of experience we have in mind, consider a user scenario generated in the City Strata project.

The creators of City Strata want to use mobile phones to allow users to tap into cinema-related memories wherever those users are in the UK (and beyond), and in the midst of their other, everyday activities.

> *Jane is walking down Whiteladies Road in Bristol. Suddenly, her phone pings to let her know she is in a hotspot from the Cinemap service she recently joined. She gets out her phone and taps "Play" on the dialog on the screen. The Cinemap app fills the screen and starts playing a short video about the old ABC cinema that used to be on that site. When it finishes, Jane flicks through a few of the related images and stories other Cinemap users have added. She notices that there is also a richer "Cinemap Walk" available for that area and adds it to her favourites for when she has a bit more time. Having spent ten minutes in the world of cinema memories, she smiles to herself and carries on to her office.*

We can pull out a few interesting features from this account: First, Jane was not actively using the Cinemap app when it detected the site of the old cinema and attracted her attention. Secondly, she happened to be in Bristol but could equally have been in Brighton or Hartlepool when a memory was detected. That's a lot of geo-tagged content to manage. Thirdly, some of that content had previously been contributed by other users. In a slightly different scenario, Jane herself might have added a fewer spoken comments and a new photo to the ABC archive.

Of course, your requirements might differ in detail from this example but it helps us to identify three broad capabilities for scalable location-aware apps:

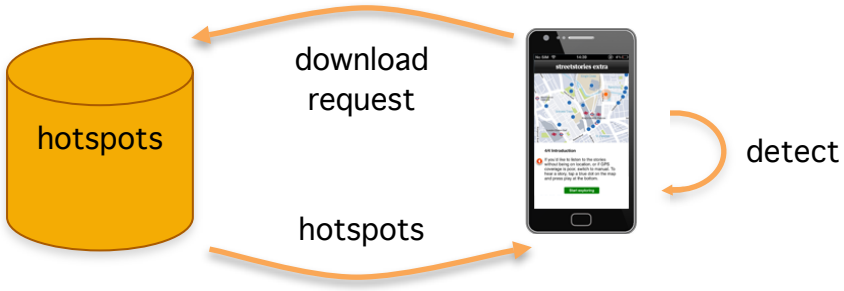
- being able to work everywhere there is potential content rather than in a restricted and pre-defined locality, including indoor locations
- access to a large geo-tagged repository of content in the cloud, including user-generated contributions
- being able to provoke user interaction at the right location even when the app is not currently running

Let's think about each of these in a little more detail.

Work everywhere

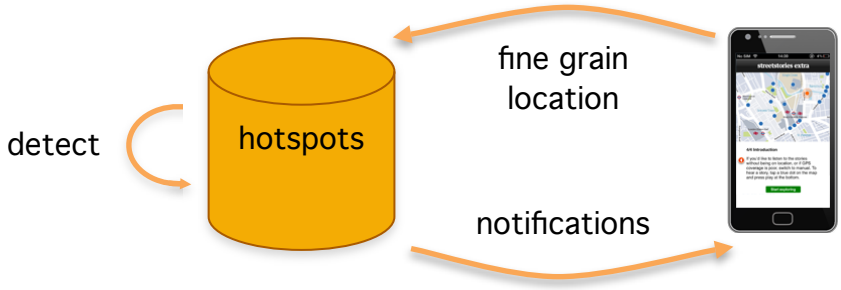
We want to be able to define lots of location hotspots – places in the world where something special should happen - for our app, and to change, add and remove hotspots even after the app is installed on users' phones. One way to achieve those aims is to store the hotspot information in the cloud rather than on the app itself. We do not even need to build an online repository ourselves - there are a number of third party cloud-based services, such as Parse, that store geo-tagged data. However, we do need to think about the simplicity, responsiveness and flexibility of the resulting relationship between the app and the online repository of hotspot information.

The simplest approach is to arrange for the app to load hotspot information from the online repository on start-up, or when the user chooses to update the current settings. The app then tracks the user's location and uses the downloaded information to detect when she has entered a hotspot.



Information-oriented apps such as festival guides often follow this strategy to reap the benefits of a simple implementation; enabling apps to operate completely independently between downloads. Another advantage is that such apps can more easily cope with network dropouts than always-connected alternatives. On the other hand, downloaded hotspot information is often incomplete and can easily go stale, perhaps compromising the user's experience. In general, this simple approach would be a good choice when hotspots do not change frequently and users can be expected to play an active role in downloading a new set of hotspots.

At the other extreme, we can invert the relationship by making the online repository rather than the app responsible for checking the user's location against hotspots.



Now, the app simply tracks the user's position and sends that information to an online service. The service runs algorithms to detect whether the user has triggered a hotspot and sends an appropriate notification back to the app. On receipt of the notification, the app renders the associated content just as the first approach does when it detected itself that a hotspot has been entered.

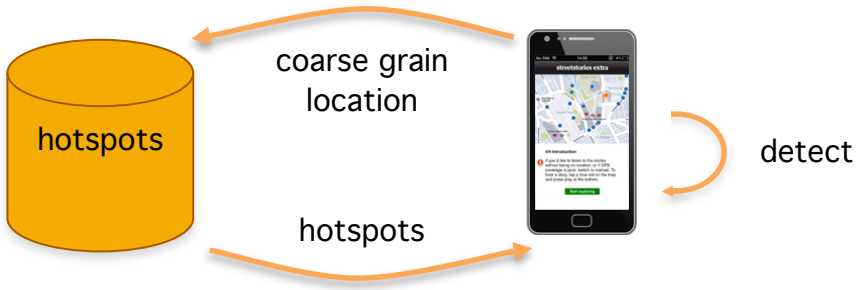
As you might expect, this approach overcomes the scale and update limitations exhibited by the simpler approach first described. The online service tests the user's position against a large and up-to-date database of hotspots. Moreover, the detection algorithms can themselves now be modified and tuned without needing to rebuild, re-submit and distribute an updated app. This approach also works well when the app is in the background - which we shall return to shortly. As a result, it is often used by location-based marketing systems, such as that provided by Urban Airship.

Naturally, there are also drawbacks, which tend to mirror the advantages of the simpler download approach. Though the app itself has a little less work to do, the overall system is likely to be more complex because it requires an online service capable of tracking and testing the location of many users in the context of many hotspots. That's certainly possible but is likely to take longer and cost more to develop than a service simply serving hotspot data. In addition, the client app's behaviour is now much more sensitive to interruptions in both the phone's network connectivity and the operation of the service

itself. And there may be users who are uncomfortable with the fine-grain online tracking of their location that this approach implies.

In practice, we prefer a combined approach in which:

- the app tracks the users location and reports coarse-grain changes to an online service
- the service responds with data on hotspots near to the user's current location
- the app then users fine-grain location tracking to detect that the user has entered a hotspot and triggers associated content



This combined approach is illustrated in the following screenshots, which shows a demo app maintaining a cache of the nearest hotspots within 1km as the user moves across the city. The green circle represents the scope of the current "radar sweep", and the orange blobs represent the hotspots detected in that sweep.

By caching hotspots in the app in this way, we can:

- handle lots of hotspots in the service and a relevant few in the app
- reflect changes made to hotspots since the user was last in that area
- minimise latency on geo-triggers due to roundtrips between the app and service
- reduce the app's dependency on comprehensive network connectivity. The app only contacts the service every kilometre or so, and has several opportunities to do so before moving out of the region of the cached hotspots
- reduce the performance requirements of the service which now needs to process location information from users much less frequently than before
- address user concerns by reporting only coarse grain locations

On a slightly different tack, there is no reason why all the hotspots in a large-scale system must be outdoors. Location tracking in mobile phones uses a combination of techniques based on the presence of cell towers, WiFi base stations and GPS. Because GPS positioning is much the most accurate of these techniques, it is often preferred for the rich location-aware apps that we are considering. However, GPS does not usually work within buildings. Tracking a user's location indoors with a similar accuracy to GPS remains a tricky issue but solutions are just beginning to emerge. We have integrated AppFurnace with WiFiSLAM to provide fast and accurate indoor location tracking for apps running on Android phones. A similar solution for iPhones must wait for Apple to provide more direct support but we can help you to experiment now with some of your user base.

Lots of geo-tagged content

If you have a lot of geo-tagged content - images, audio and video - associated with hotspots then it also makes sense to store the content files online rather than in a mobile app. The issue then, of course, is that the content must be retrieved from the online repository before it can be rendered, for example when a user enters an associated hotspot. That means an inevitable delay while the content data is brought over the network. A streaming server will allow some content to be rendered before data retrieval is complete but even so, the delay can still be noticeable over typical 3G conditions. Where connectivity is poor or non-existent, the content data may never arrive.

One way to mitigate this problem is to fetch content data in anticipation of its use. For example, we might retrieve the content data associated with the five nearest hotspots so that it is immediately available should the user enter one of them. Such pre-emptive data retrieval can be done quietly in the background without disturbing the user experience. However, the inevitable cost of a pre-emptive approach is that the app will occasionally (or often) retrieve data that is not then used. Moreover, the need to manage the app's cache of content data within its allocated storage allowance on a phone means that some data may be discarded and retrieved more than once. For some users, that may mean extra data charges. In the worse case, the data limit on the user's tariff might be breached, especially when using rich media such as video and audio. That would seriously degrade the user's experience.

So, pre-emptive data retrieval is potentially advantageous but needs to be carefully designed. In practice, we would tend to use a hybrid approach in which text and small images are pre-emptively fetched to reduce latency while richer media is fetched on demand. And control should certainly be given to the user to prohibit pre-emptive data retrieval, perhaps to the extent of requiring explicit permission from the user before using the technique at all.

Another common requirement of the kind of service we are considering is that users can themselves contribute geo-tagged content when out and about. The first, obvious, implication is that the app must be able to capture the content directly on a phone. That's not something yet available in the public version of the AppFurnace tools (other than for text input), but the version we use internally is able to create apps that allow the user to take photos and record audio and video.

A more subtle issue concerns what happens to the content so captured. The simplest approach is to immediately upload the content data to the online service. But what if the phone is not currently connected to the network, or if the user would prefer to wait until they have a free WiFi connection rather than use up their 3G data tariff? That implies a need for managing captured content on the phone for later uploading, perhaps opportunistically when an appropriate network connection is detected. You might also want the user to be able to delete content they have added, or amend it. None of these requirements raise insurmountable problems, of course, and they have often been tackled in other settings, but they are a reminder of the careful thought often needed in the design of a service-connected app.

Background monitoring

People use their mobile phones for many purposes and it is, of course, unreasonable to expect that they will always keep our app on-screen in the hope that their changing location will trigger an interesting or useful experience. Rather, we would like an app that continues to monitor the user's location even when it is not active and notifies her when she moves into a spatial hotspot with interesting content. Of course, an app should only operate like that with the user's permission - partly because seeking permission is always the right way to behave but also because there are specific issues with background location tracking.

Remember that mobile phones use a combination of three methods to determine their (and hence the user's) location:

- GPS - the phone receives signals from the worldwide network of Global Positioning Satellites and works out its location every second to around 10m under good conditions (for example in the open air and away from tall buildings).
- WiFi - the phone detects nearby WiFi base stations and compares the resulting pattern to a database of known locations. Accuracies of around 100m might be expected in urban areas with lots of WiFi.
- Cell Tower - very similar to the WiFi method except that the phone uses the cell transmitters at the heart of their particular mobile network. Because these are much more sparse than WiFi base stations, the achievable accuracy is around 1000m.

Broadly speaking, GPS is an order of magnitude more accurate than WiFi and two orders better than cellular positioning, so why would the other two approaches ever be used? The answer lies in their differing power requirements. GPS tracking requires special circuitry in the phone that can drain its battery in just a few hours. For a location-aware app that a user has chosen to run on-screen in "foreground" mode, that's fine. The user is

likely to use the app only for a limited time, and can monitor how the battery is faring and quit the app if necessary.

However, background location tracking is intended to operate without the user's attention until a notable location is detected. While an app can use GPS to track the user's location in the background on a mobile phone, there is a real risk that it might cause the phone to run out of power without the responsible app appearing to do anything at all. That is very unlikely to recommend the app to the user or their friends.

Cellular positioning, on the other hand, requires no extra power from the battery. Phones must connect to cell towers anyway to maintain connection to the network, and the built-in location tracking algorithms piggy-back on this core activity. An app that used just cellular positioning for background location tracking would therefore be a much better citizen with regard to battery life and the user's overall experience. However, there is a cost in both reduced accuracy and responsiveness.

For example, iOS makes it possible to use cellular positioning to wake up dormant apps briefly whenever an iPhone detects that it has moved a significant distance or entered a spatial hotspot registered with the operating system. But, "significant" in this case might mean a kilometer or so, and the spatial hotspots need to be hundreds of meters in diameter or more to stand a realistic chance of being triggered. That may be just too coarse grained for an app that is intended, say, to tell you about the rich history of a demolished cinema as you stand outside where its grand entrance used to be.

So, at the two extremes we seem to be faced with a choice between a background location tracking approach that is accurate but power-hungry, or energy efficient but coarse grained. The resolution, as often the case, may be to use a hybrid, multi-scale approach that combines the best qualities of both cellular positioning and GPS (and also leveraging the intermediate characteristics of WiFi positioning where appropriate). For example, we might distinguish between three background tracking modes:

- orienting mode - in which the objective is to detect that the user has entered a large geographical area that might contain locations of interest to the user. For example, our example app might need only to detect that the user has entered central Bristol where located cinema memories are known to exist. Cellular position works well in this mode.
- monitoring mode - in which the app begins to monitor the user's location more closely but still with a strong emphasis on power saving. WiFi positioning has a role to play here. The aim is to detect that the user is sufficiently close to target hotspots to switch into the triggering mode.
- triggering mode - in which the app temporarily uses fine grain positioning based on GPS to detect the user's arrival at a location of interest and to trigger the associated content.

Getting this mixture right for any particular app is likely to be subtle and app specific. But it does offer a way for an app to deliver the user experience we desire without draining the battery.

Earlier, we mentioned in passing that background apps can be woken up briefly to handle changes in location. The word "briefly" is important here. Another way for an app to be a bad citizen would be to grab all of the phone's resources (screen, CPU, connectivity) while the user really wants to do something else. Apple is sufficiently concerned about this risk that it deliberately limits what an iPhone app can do when it is woken up in the background. That means the app should not try to render content files directly on being woken up to discover the user is in a hotspot. Instead, it should raise a notification to the user requiring an explicit response to invoke the content, and go back to sleep. Of course, that approach implies that the app knows the locations of the local hotspots and their associated content, which takes us back to an earlier discussion. In practice, apps may simply use their brief time awake to report their location to an online service that decides whether or not to post a notification back to the user.

Closing thoughts

As that last sentence suggests, the three issues discussed in this white paper are really all interlinked. A strong requirement for an app to respond to hotspots even when it is inactive pushes us towards online detection, at least on iOS. But online detection implies that the hotspot information is also stored online. In turn, that suggests that we keep associated content online too. Now we have to think about how to retrieve that content to balance latency and data transfer costs. And all of those decisions may be influenced strongly if it becomes imperative for the app to operate indoors.

Scaling a location-aware app to support large amounts of geo-coded content goes far beyond the design of a standalone app. A systems approach is needed and there are subtle architectural issues to consider. Implementation is not always straightforward but it is possible, especially when the specific requirements of a particular user experience are taken into account. We in Calvium would love to help you take your ideas for a scalable location-aware app further. Please contact us using the details overleaf.

Acknowledgements

The City Strata project was a collaboration between Calvium, Peter Insole (Bristol City Council) and Dr Charlotte Crofts (University of the West of England). We would like to thank the REACT hub for supporting the investigations underpinning this paper as part of the Heritage Sandbox programme. Funded by the Arts and Humanities Research Council (AHRC), REACT (Research and Enterprise in Arts and Creative Technology) is one of four UK Knowledge Exchange Hubs for the Creative Economy and is a collaboration between UWE Bristol (the University of the West of England), Watershed and the Universities of Bath, Bristol, Cardiff and Exeter.

Calvium provides mobile apps and software services to companies, arts organizations, universities and other customers. Founded in 2009, the Calvium team is passionate about making technology accessible to the creative industries through its online AppFurnace tools, training workshops and custom mobile app development. We have a wealth of experience helping clients such as the Royal Shakespeare Company, Historic Royal Palaces, and The Guardian to develop exciting, innovative applications that connect with audiences in powerful and unexpected ways. To find out more, visit the Calvium web site or contact us using the details below.

Website	www.calvium.com
Email	hello@calvium.com
Tel.	+44 (0) 117 370 875