Solving App Performance

Six Tricky App Performance Questions Answered

Contents

Introduction

Mobile devices have become as essential as house keys and wallets, in some cases even replacing them. This reliance on mobility, however, causes a new set of challenges for mobile application providers. End users are impatient, they want apps to perform at desktop or even real-time speeds. If an application does not, there are millions of other apps available to take its place.

Due to users' reliance and expectations, there is a growing pressure to distribute more and more data both from servers to client devices, and from those client devices back to the server to provide a well-perceived experience. However, as the amount of data produced and consumed grows, it impacts performance and efficiency – two criteria paramount to the success of mobile applications. As a result, application developers are now forced to ask themselves:

- What does it mean for an application to be performant?
- What happens to the application under fluctuations in load?
- How does the application handle explosive growth?

This paper focuses on each of these questions and more, delving into how best to look at applications and where typical performance issues arise. By the conclusion, application developers should be able to effectively evaluate their applications with an eye to performance.

Performance, a Complex Word

How to define performance, based on end users, is tricky. If 100 end users are asked what makes an application performant, just as many replies might be given. Performance first, and foremost, is based almost entirely on end user or consumer perception. This makes the performance metrics of any device or application very difficult to evaluate using standard metrics. End users might use criteria like:

Performance first, and foremost, is based almost entirely on end user or consumer perception.

Initial load (startup) times

- How quickly does the app load?
- How quickly can I start using the app?

Responsive behavior

- When I touch or click, does the app respond quickly?
- Can I easily navigate around the app to do things?

However, there are several other areas of performance that can be much more quantitatively measured. Quantitative measurements allow businesses to set baselines, show improvements and definitively know when development cycles have been successful in increasing performance. Criteria that can easily be measured to show capability and performance gains are:

Quantitative measurements allow businesses to set baselines, show improvements and definitively know when development cycles have been successful in increasing performance.

Network capabilities

- How does the app behave on high latency networks?
- How does the app behave on unstable networks?

Resource utilization

- Does the app behave normally on less than ideal hardware?
- Does the app excessively use processing power and/or battery?
- Does the app needlessly consume data, or use excessive amounts of data for a standard data plan?

Load fluctuation

- How does the app handle temporary spikes in consumption?
- If a user requests more information, is his or her experience affected?
- Is the user experience affected if the app user base grows by a factor of 1000?

Using the above as app criteria, developers can get apps performing as they should.

Six Reasons Why Mobile Application Performance is Difficult

The problem for developers is that the performance criteria identified comes with existing challenges. This is not meant to be a comprehensive set of all challenges, just some of the most prolific issues that face application developers.

1. **End user perception reigns supreme.** It doesn't matter if an application needs to load no data or reams of data on load, end users don't care. What end users care about is **initial load times**. In fact, most users expect their mobile devices to now perform at or close to the same speeds as their computer. **How can application developers make applications that operate at desktop speeds?**
2. **One chance to show value and capability.** Once an application has loaded, it has one chance to convince a user that it is performant and provides **responsive behavior**. Studies show that almost 80% of users will delete an app after using it for the first time, simply because of bad design, usability, load time and crashes. Applications that are sluggish are worse than applications that take time to load. Sluggish behavior leads to incorrect button touches/clicks and even more wasted time. **How can application developers make applications that respond quickly to user feedback?**
3. **Global networks are unreliable.** Applications are evaluated on **network capabilities**. Probably the biggest issue for any application that requires network connectivity is the quality of the network the end user has available. Advances in technology have happened so rapidly that mobile carriers are struggling to keep up. Software capabilities have far outstripped the physical networks they need to run on, and this problem is borne not by the end user, but by the application developer. **How can application developers take unreliable networks and still provide a performant end user experience?**
4. **Devices have limited power and performance capabilities.** Networks are not the only hardware component that is struggling to keep up with software. Physical devices are limiting factors when it comes to application capabilities. As the majority of devices become mobile, a distinct reliance upon battery power makes r**esource utilization** a key concern for any application developer. Process intensive computations can affect responsive behavior of an application, increasing battery consumption of the device. Applications that heavily consume battery life can now easily be identified by mobile operating systems, rapidly showing users non-performant applications. In fact, poorly written applications can zap up to 30-40% of a user's battery. **How can application developers write resource efficient applications?**
5. **Data consumption is precious to both providers and consumers.** Another key factor in resource utilization is data consumption. Due to the aging mobile network and the explosive growth of mobile devices, carriers are all carefully watching data consumed by end users, and punishing them greatly for excess usage. As with battery life, this is important enough that mobile operating systems should show which applications are eating a data plan. Add in an increasing user demand for real-time data, and application developers find themselves between a rock (end users) and a hard place (user data plans). **How can application developers write real-time, data efficient applications?**
6. **Usage patterns have much larger peaks.** Most real-time applications face their first critical performance issues during periods of extreme **load fluctuation**. Every application developer's nightmare is that some big event will happen and the load will cause all users to suffer performance problems. Coca-Cola experienced this very problem during a Super Bowl ad campaign that was so successful, it caused extreme load on its servers and took them down completely. Not only was this a Fortune 500 company with dedicated staff and user bases that make most application developers drool, but this problem happened on a desktop application! **How can application developers handle extreme spikes in user load?**

With all of these issues clearly identified, it is time to examine how best to approach the problem of performance as it relates to a mobile application. This covers the most common issues faced when discussing application performance.

Six Ways to Improve Mobile Application Performance

App performance can be broken into six manageable questions that development teams can tackle independently. These six will be examined to determine how to best solve these common problems. There are several solutions to each of these, but the end result should be that the end user experience is positive. While each solution can be handled independently, there are often technologies or features that will help solve several of these issues at once. Each solution will focus on business concepts, architectural ideas and even talk about some specific technologies in the space.

1. How can application developers make applications that operate at desktop speeds?

This problem has plagued developers of all applications for several years. Initial implementations provided HTML in full for the browser to render. A change to any small part of the GUI would then be delivered as a new, full HTML page. As pages became larger, this proved to be more and more inefficient, leading developers to focus on how to only update the components and data that had changed. First addressed by Microsoft in 1996 with the iframe tag, it saw an updated rendition as ActiveX in 1998. In 2005, over a year after it was first used by Google and Kayak, the term AJAX was coined. The focus of updating only the components that have changed or need updating makes logical sense, especially as applications and sites become more and more complicated.

As technology progresses, these efforts have been transformed again to utilize WebSockets, which unlike AJAX (a HTTP request) allows for full-duplex communication. WebSockets have been specified to resemble HTTP so both HTTP and WebSocket connections can be made on the same port, with the caveat that the specific fields, and what follows the handshake, do not conform to HTTP. WebSockets also allows data to be multiplexed across several streams simultaneously providing asynchronous update capabilities for each component that needs to be changed.

- *Show cached or static content to the user quickly.*
- *Load dynamic content in the background.*
- *Consider using WebSockets to transmit real-time data.*

TIP

With these capabilities and technologies, developers can create a simple static landing page that can be updated behind the scenes. This is simply a trick using something magicians are all too familiar with, sleight of hand. The end user sees a page that contains relevant information and that information is then updated as it is pulled down from the network. Mobile applications have the distinct advantage of built-in storage and cache that are local to a user, which allows developers to store more recent dynamic information that can be quickly accessed by the user. This information can then be updated in the background, or more recent data can be pulled down once the application is loaded.

2. How can application developers make applications that respond quickly to user feedback?

As with any piece of software, the more native the code, the cleaner it will typically run. Several studies have shown that mobile sites (as opposed to true native applications) have more negative user experiences than fully native applications. The problem that most teams run into when applications need to be true applications, and not mobile sites, is dealing with the variety of tools, languages and frameworks required to develop for the mobile operating systems available.

- *Go native with applications; mobile sites are not cutting it.*
- *Several tools are available to help ease multi-platform development.*
- *Test, Test, Test and leave the functional testing to automation so QA teams can focus on performance and experience.*

TIP

To combat the issue of several mobile operating systems, tools such as Cordova, Qt, and PhoneGap (just to name a few) allow development teams to develop applications one time and deploy them to several operating systems. Many application developers will find this easier to use than developing native applications, but these should still be considered if high performing applications are key. These tools are an abstraction of native functionality, and thus often not as performant as a truly well developed native application.

As with any application, none of these abstractions are a replacement for a solid test scenario, both automated and manual. For automated testing, tools such as Appium provide an open-source solution to help ensure functionality, and good quality assurances leaving testing teams to focus on performance and experience. Finally, adding analytics and metrics to mobile applications can help understand user journeys. This knowledge can provide insight into how end users utilize the application and where users exit the application. If user journeys suddenly end on a set or a singular scenario, chances are it needs closer attention by the development team.

3. How can application developers take unreliable networks and still provide a performant end user experience?

When Google started to examine how to make the web faster, it started an experiment called SPDY. In a white paper, Google found that typical request/response headers were 700-800 bytes, but could be a lot bigger depending on the application. Not only were the headers rather large, but they found there were several headers that were redundant, essentially anything that was generally static (User-Agent, Accept, Host, etc.) that still were sent with every request.

Rather than use a standard request/response model, application developers can begin to combat aged network latency by using publish/subscribe models. This model doesn't require the client application to continuously poll the server, saving the request/response overheads. In fact, Push Technology's Diffusion product uses a keep-alive of approximately 25 bytes. This means your standard request/response will consume 1400-1600 bytes per request/response lifecycle, whereas Diffusion would consume 25 bytes every 90 seconds. In real-time applications that need to poll on a schedule of 10 seconds (or less in some cases) this can mean the difference of 576KB vs 1 KB for a single client over an hour. That is a saving of 575 KB in the header alone, which can go a long way to help manage high latency networks.

Application developers also need to pay careful consideration to how their applications handle unstable networks. Mobile devices mean that users are typically just that, mobile. While applications are utilized, connectivity will change as the user moves around. Typically when using a publish/subscribe model, the data expensive actions happen when establishing the connection and setting up subscriptions. Handling connection loss at scale, especially with an eye to performance, can be complex for application developers. This makes finding a model or third party software solution that can handle highly latent or unstable networks without causing the connection to be lost important to teams, but the benefits to mobile application performance can be profound.

- *Rid yourself of request/response in favor of publish/subscribe.*
- *Be sure to consider constant changes in network quality and handle them accordingly.*

TIP

4. How can application developers write resource efficient applications?

Resource efficiency begins with the hardware and battery life of a mobile device. With an almost infinite variation in device hardware and battery life, especially since these components do age and change in capability, this is a very difficult problem to tackle for real-time applications. Fortunately for development teams, cloud solutions have come to the rescue. Utilization of cloud technologies allows process intensive calculations to be offloaded to the cloud, and then the results can be returned to the application to be rendered to the user.

This solution does come with its own set of problems, namely data consumption. As computation is offloaded to the cloud, data needs to be sent from the mobile device to the cloud servers, and then needs to be returned to the device once a result has been rendered. By solving the first component of resource performance, hardware and battery life, the second component of resource performance, data consumption, has grown considerably.

- *Offload heavy calculations to a cloud solution.*
- *Cloud has the distinct disadvantage of creating a larger data efficiency problem.*

TIP

5. How can application developers write real-time, data efficient applications?

Presuming application developers have moved to a publish/subscribe model, especially over WebSockets, they have already started to manage data concerns. However, the publish/subscribe model typically is focused on a single direction of data flow. In the cloud-offloading example above, data needs to be sent and received from the server by the client. This typically requires a more advanced message broker that can handle a bi-directional flow of data.

Data resource management should focus on only publishing when data has changed. Be careful of brokers like ActiveMQ, who publish every message sent to them. Ensure that server applications are intelligent enough to understand if content has changed and publish it accordingly. When data is published, also ensure that you make use of intelligent deltas, or only publish the data that has changed. As earlier calculations have shown, a few bytes of data savings can add up very quickly over time.

Rather than utilize ActiveMQ or RabbitMQ, application developers can utilize Diffusion to be their message broker. Diffusion utilizes a publish/subscribe model that is stateful, allows for bi-directional messaging and handles all of the delta calculations automatically. Diffusion also naturally handles WebSockets, and the intelligent degradation of communication protocols until the most efficient protocol is utilized for the client. This allows clients to connect using the most efficient method, and send/receive as little data as possible. This functionality means that Diffusion pushes the right data to the right user at the right time, without forcing application developers to understand and/or write their own versions of intelligent deltas, client management or WebSockets.

- *Utilize publish/subscribe models over WebSockets.*
- *Focus on bi-directional data-flow solutions.*
- *Only publish when data has changed, and what data has changed.*

TIP

6. How can application developers handle extreme spikes in user load?

Once a mobile application has been tuned to a level of high performance, released into the wild and starts to be used, development teams typically breathe a sigh of relief. However, today's world is one of high connectivity and social media, meaning this should be the scariest time of any application's lifecycle. A truly performant application with sudden and increasing user adoption can lead to a rather large scaling issue. Solving this problem before it becomes a problem is really the capstone to any high performing mobile application.

To use a real world example, imagine you came up with the next Uber, where you went from a concept, to an app in a single city, to a global name, all in a very short timeframe. These issues of scale can be managed much more easily today than ever before, simply using cloud technologies and services. Cloud technology like AWS provides a simple way to start small and grow with a user base, but comes at a cost. Focusing on an efficient server side solution is as important as the focus that is paid to the mobile application itself.

Even big companies struggle with this problem, not due to user growth, but simply due to additional mobile load spikes. During Black Friday weekend, Best Buy's website crashed for 90 minutes. When asked about this, the company said:

> *"A concentrated spike in mobile traffic triggered issues that led us to shut down BestBuy.com in order to take proactive measures to restore full performance."*

Mobile application developers need to look not just at how their application can grow, but how it can handle concentrated numbers of users at the same time.

Building mobile applications that can provide a consistent experience with high availability assurances during spikes in user adoption is a challenge when evaluating a single server. Utilization of cloud technologies means this needs to be evaluated on a distributed system, potentially with global distribution as well. Server side applications should be able to rapidly connect and distribute data to several thousand clients quickly and easily. Finally, this scale should not strain data providers to a point of failure. While a server side application might scale accordingly, will the data provider scale just as well? This evaluation should be completed against a full, distributed stack, not just a singular instance or component. As a matter of reference, a single instance of Diffusion can scale to almost 90,000 concurrent clients **without affecting the latency of each connection** and provides a single source for your application data provider to update.

During testing, do not forget to utilize tools like JMeter to issue requests against server side applications. If the mobile application is utilizing a publish/subscribe model or middleware, ensure there is access to some kind of load/performance testing tool or suite.

When integrating Diffusion with clients, Push Technology's Professional Services team utilizes a suite of tools that have been developed specifically to test load and performance against real application data, not just benchmark or test data. Whichever solution works best for development teams, ensure that there is time to learn and/or develop test applications.

- *Utilize cloud technologies to scale backend services rapidly.*
- *Understand connection and load scale for all components of an infrastructure.*
- *Performance and load test servers to understand impacts of end user or utilization spikes.*

TIP

How Diffusion Helps

To deliver improved application performance, organizations must have the ability to solve issues that are inherent when distributing data. Diffusion from Push Technology addresses all of the issues outlined above by intelligently distributing distributing the data required to deliver rich applications. Businesses benefit from higher application performance, faster apps so opportunities are not missed and the ability to scale so, regardless of concurrent connections, the organization can meet end-user performance needs. With Diffusion, the application experience is massively improved.

Conclusion

Performance is a complex subject to discuss, especially given its largely qualitative nature. However, main performance issues can quickly be boiled down to focused questions, allowing for mobile application developers to quickly start to analyze and improve their products based on standard criteria. Are there exceptions to these guidelines? Of course. But overall, mobile application developers should be able to evaluate these challenges and questions and solve them to provide end users a high performing experience.

- Perception is king.
- Focus on cached or static content at load time.
- Load dynamic or updated content in the background.
- Consider using WebSockets to transmit real-time data.

- Responsiveness to user feedback
  - Native applications provide better experiences.
  - Handle multi-platform development cleanly.
  - Mixing automated and manual testing provides optimal utilization of test resources.
- Managing aged networks and inconsistent network connectivity
  - Move to publish/subscribe models over request/response.
  - Ensure applications can manage intermittent connectivity and changes to network capabilities.
- Resource efficient applications
  - Utilize cloud solutions for heavy calculations and processor intensive actions.
  - Be aware of data increases when offloading resources to cloud solution.
- Real-time data efficient applications
  - Focus on bi-directional publish/subscribe solutions using WebSockets.
  - Only publish data that has changed including intelligent delta management.
- Extreme load spikes
  - Rely on the cloud for scale, but be sure to watch costs.
  - Understand connection and load scale for all components of an infrastructure.
  - Performance testing should never be an optional part of testing cycles.

Careful analyses of these measurements allow teams to release highly performant applications, and continued work on improvement should keep end users happy and active. As users continue to see performance gains, they will share their experiences, growing user bases and making an application successful in the market today.

About Push Technology

We make the Internet work for our mobile-obsessed, everything-connected world. Leading brands like 888 Holdings, DAB Bank, IBM, and William Hill leverage our technology to power applications critical to revenue growth, customer engagement, and business operations. Learn how to deliver apps at scale and speed at **www.pushtechnology.com**