Performance Issues in Mobile Computing

CSCI 5708 / 4176

Mobile devices face many limitations when compared against desktop computers

- Less powerful CPUs
- Less storage; both volatile (core) memory and non-volatile long-term memory
- Decreased power availability (battery-powered)
- Limited network bandwidth
- Higher data costs
- Higher network latency
- Less skilled users
- Decreased screen space, no keyboard
- Greater variety of operational environments (e.g., buses, planes, outdoors)

The Challenge:

Mobile devices are small and low powered.

Users:

- Short attention spans
- Limited patience for slow apps
- Limited time to wait for an app to complete a task
- Expectations of good (fast) interactions with the app
- Expectations of phone being “on” all the time

Performance is Important!

If an application takes too long to load or perform a task, the user will often abandon, exit, or delete the application.

Mobile devices will continue to shrink.

Examples:

- *Watches, Sensor tags*

Thus, computing resources will continue to scale down.

What is the impact of further scaling down of devices?

And, WHAT REALLY MATTERS TO USERS?

Performance Concerns

We are primarily interested in performance concerns that matter to the **user**:

- **Latency** : The delay between the start and end of an operation.
  - ***Network/Communication*** latency is delay due to network communication.
  - ***CPU/Computation*** latency is delay due to CPU activities.
- **Throughput or Bandwidth** : The amount of data that can be processed or transmitted/received per unit time.
- **Memory/Storage Usage** : The amount of memory or storage needed to run the application or perform a task.
- And less noticeably, **Power Usage** : The amount of power needed to perform a task.

Metrics that Matter Most

- The most noticeable performance metric, from a user perspective, is latency. Note: Users tend to blame the network for any/all latency.

"*the network is very slow today!*"

- Throughput/bandwidth also affect perceived latency.

- Power and storage use are noticed but are not immediate show-stoppers. Users notice these when they don't have enough (then they perhaps do something).

Desktop vs Mobile

What's the difference between the mobile and desktop/laptop world?

	Desktop / Laptop	Mobile
CPU	3Ghz 100W, 240,000 Dhrystone MIPS	1.85-2.2 GHz ~0.5W, 13000 Dhrystone MIPS -- Passive cooling is a major factor
RAM	4 – 32 GB	2 – 8 GB
Power	Wall plug or 50-100Wh battery	5Wh Battery
Storage	512MB - 2+ TB	32 – 256 GB
Network	10/100/1000 Ethernet, Wifi 802.11ac (50 MB/s)	Wifi 802.11ac (50MB/s) Cellular (3-10 MB/s on average) -- Longer transmission distances

- Mobile devices: 10x slower, 1/10 power capacity,1/10 volatile storage, 1/10 persistent storage
- Higher communication costs than desktops/laptops
- Developers need to mask these performance gaps to meet user expectations.

Trade-Offs

- *A trade-off is a balance between **two desirable but incompatible goals**.*
- Resources we can trade-off:
  - Power
  - CPU
  - Memory
  - Storage
  - Bandwidth (how much we use)
  - User latency (how long a user waits)
  - Developer time
- **We can conserve some resources by making trade-offs.**

  E.g., Cheap, fast, or good; choose two.
- To conserve one resource, we expend other resources.

More Observations

All resources, but two, are device limited.

Developer time is not dependent on the device.

Network bandwidth/latency are network dependent.

User latency depends on (the lack of) most resources.

Most resources are correlated to power use.

Common Trade-Offs

The Human Trade-off:

Spend more developer time to develop less resource hungry applications.

Other common trade-offs:

CPU for memory/storage

(memory is usually cheaper)

Memory/storage for CPU

CPU/Memory for bandwidth

Power for all other resources

Reduced functionality for Power

Computational Latency

Definition: ***How long a user waits for an app to finish computing a request.***

- Predominantly depends on the CPU and memory/storage system.
  - Data must be moved from storage to memory before it can be used.
  - Some calculations are just slow (e.g., approximations for NP Complete problems)
  - More computation implies user waits longer for action to complete.

Network Latency

Definition: ***How long a user (application) waits between sending a request over a network and receiving a response.***

- Depends on the network being used and the remote host.
  - User/app has little control over network or remote host.
  - Cellular networks have a high latency:
    - 111ms on average in US (4G)
    - 150ms on average in US (3G)
    - Not uncommon to have 500ms latency
  - Must consider – the *time for the request to be received* by the server, and the *time taken to service the request*.
  - Requests can sometimes take a second or more to be serviced.

Limited Throughput (Bandwidth)

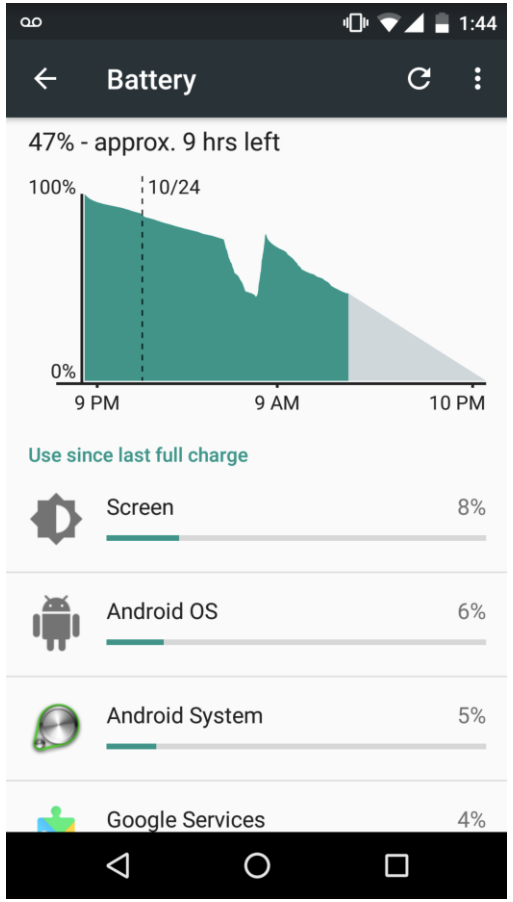
Definition: ***The amount of data (number of bytes) that a user can send/receive per second.***

- Depends on the network, with little control by the user.
- In many places (especially in less populated regions) cellular networks do not have a lot of bandwidth
  - E.g., In some places it is still 4KB/s
- Also, in some places data costs cause users to limit/avoid bandwidth usage

If using a networked server introduces latency or bandwidth concerns, does it still make sense to delegate servers to manage computation and/or other aspects of your app?

Power Usage

- The more power an app uses, the shorter the time between recharges.
- In the best case, this is just an annoyance.
- In the worst case, the phone is killed half-way through the day, possibly with no possibility for recharge.
- **Users only notice when their battery dies.**
- **Rarely is it linked to a specific application.**



Tradeoffs

- Compression: Less data/memory, more CPU/power to compress, decompress
- Caching results: Less CPU to recalculate, more memory to store results
- Pre-downloading: May reduce network latency but increases bandwidth and memory usage
- Local computation: Saves on using network for data and avoids network latency, but uses power and creates CPU latency
- Remote computation: Servers have more CPU and storage so saves on local storage and CPU, but uses data and creates network latency
- Cloud storage: reduces local storage but introduces network latency, bandwidth issues
- Distributed processing? Offload computation to paired devices (e.g., iPhone and Apple Watch)

User Control and Freedom

- Do not assume that all members of the target audience are the same!
- Different users may have different needs or preferences.
- Giving users control in settings may be sufficient.
- Don’t decide everything for the users or think that as a developer “you know better” …
- Allow users to turn features off (e.g., sound), reduce brightness, decrease accuracy, use compression, etc.
- Many tradeoffs introduce additional issues (e.g., privacy concerns with cloud storage).

So, What Can We Do?

- Fundamentally, we cannot run away from the laws of physics.

- Our apps have to contend with
  - Slower CPUs
  - Networks with poor connectivity, lower bandwidth, and higher latency
  - Limited power

- ***We need to mask or manage these issues to deliver the best possible user experience.***

- *E.g., spend a considerable amount of time testing your app prototypes, make sure that they're efficient and make it easier to optimise resources.*

Less is (usually) More.

Evaluation

- After making trade-offs, we need to evaluate their effectiveness.
- How do we perform this evaluation?

Evaluation …

- Survey the target users.
- Measurement of resource use or latency in development settings.
- Monitoring of performance with actual users.

Anyone can program a mobile application;

But can you:

- Ensure it has a fast response, doesn't "hog" resources, manages bandwidth effectively, and handles network latency issues?
- Manage privacy, security, and legal concerns?
- Make it usable, accessible, and learnable?
- Test it and perform effective quality assurance (QA).
- Program it in a way that permits you to maintain it, migrate/port it, and keep it functioning correctly?
- Know your target audience and give them what they need (and not what you or they think they want).
- Ensure that it is downloaded and used (i.e., adopted and retained).
- And lastly … make some money doing this?

If you can, then you are a
Successful Mobile Application Developer!


Uniqueness
User Friendly
Rich Features
Flawless Coding
Powerful Marketing
Mobile Application Development