Mobile and Ubiquitous Computing
2021/22
MEIC/METI - Alameda & Tagus

Cyberforaging

Introduction

Should apps run entirely on mobile devices ?

Motivation – how to partition

- Mobile applications increasingly pervade our daily lives

- Innovation in computing hardware has continually provided mobile users with more computational resources in lighter and smaller form factors:
  - laptops have replaced personal computers, and
  - are now in turn being replaced by tablets and smart phones

- The portability of current mobile devices satisfies this demand by allowing their users to run applications anywhere at any time

- However:
  - most mobile applications do not run in isolation on the mobile device
  - they access data and computational resources in the cloud via wireless (e.g. cellular base stations and WiFi access points)

- Wired infrastructure components such as compute servers and data stores provide more computational resources than a mobile device:
  - they are not limited by stringent size, weight, and battery energy constraints

"How should one best partition applications across mobile platforms and fixed infrastructure?"

Motivation - static vs dynamic

- Static partitioning fails to account for the variability inherent in mobile and pervasive computing environments:
  - wireless **network quality can change** by one or more orders of magnitude as users move
  - mobile devices **may become disconnected** from the fixed infrastructure
  - shared computational **resources in the cloud may become overtaxed**
  - nearby computational **resources may become available** for opportunistic usage
  - **resources** available to different mobile devices **can vary substantially** from platform to platform
  - **battery/energy/cellular bandwidth are limited resources** and may need to be consumed within a budget
  - application **demand** and user **workloads** may **change over time**

- There is no “always-best” static partitioning of application functionality

- Thus, any static partitioning will inevitably lead to:
  - a poor user experience when the actual usage environment varies from that envisioned by the developer

Cyberforaging – partition, migration, replication

- It enables the seamless mobility of data and computation
- Users are given the illusion that their applications run entirely locally on their mobile devices
- This illusion preserves the pervasive, always available access to applications and data that has driven the growth of mobile computing
- Behind this illusion, however:
  - computation and data may be replicated and migrated between the mobile computer and fixed infrastructure

What is Partitioning

Given a specific application state and a specific computational environment, which portions of the application should run on the mobile computer and which should run on remote infrastructure?

- Dynamics depend on:
  - the application state may depend on specific inputs and the current location of data used in the computation
  - the mobile environment (server load, network bandwidth, etc.) is constantly changing

- Partitioning decision:
  - **objectives** that cyber foraging systems attempt to achieve through application partitioning and the **metrics** used to quantify those objectives
  - different approaches to enumerating the **set of candidate partitions**
  - strategies used to **select** one of those candidates
  - strategies to **estimate resource supply and demand**
  - how cyber foraging **recovers from failures** that occur during partitioned execution
  - which **characteristics** make applications particularly well suited or poorly suited for cyber foraging

What are the benefits of partitioning apps?

Potential Benefits

- There are many reasons why an application may benefit from ***executing a computation*** or ***storing data*** on remote infrastructure instead of or in addition to the mobile computer on which it is currently executing:
  - better performance, less battery used, data fidelity

- There is a fundamental performance gap between mobile and fixed computers:
  - even as computing power has increased rapidly over the past few decades, the computational resources available to mobile devices have substantially lagged behind their fixed counterparts

Year	Representative server		Representative handheld	
	Processor	Speed	Device	Speed
1997	Pentium II	266 MHz	PalmPilot	16 MHz
2002	Itanium	1 GHz	Blackberry 5810	133 MHz
2007	Core 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2012	Xeon E3	14 GHz (2x4 cores)	Samsung Galaxy 3S	3.2 GHz (2 cores)

- Comparing the processing power of representative computer systems in five-year increments from 1997–2012:
  - as a rough estimate, processing power is given by the **product of core count and clock speed**
  - although both server and mobile computers both show rapid growth in processing power, the **performance gap between the two remains substantia**l for every time period examined

Benefit - performance

- Due to the gap between mobile and infrastructure processing power:
  - **compute-intensive** applications can execute much faster on remote infrastructure than on mobile devices
- On the other hand:
  - **interactive activities** that demand few computational resources may execute almost as fast on a mobile computer as they do on a server
- Further, performance is not impacted solely by processor speed:
  - remote infrastructure may offer **more memory and storage**,
  - the ability to **parallelize computation** across multiple cores and servers, or
  - better **network connectivity**

Benefit - battery

- Designers strive to make mobile computing systems as energy-efficient as possible, for example:
  - employing hardware power-saving modes, or by
  - reducing the scope or quality of activities performed by mobile applications

- While these measures are essential to extending battery lifetime, they also:
  - noticeably degrade the mobile user experience
  - applications take longer to perform interactive activities and produce lower-quality results

- Use of fixed infrastructure is an attractive alternative for designers:
  - by **offloading computation or data storage** from a battery-powered computer to a remote computer with wall power,
  - the operational lifetime of the battery-powered mobile computer can sometimes be extended
  - without degrading the user experience

Benefit - fidelity

- Fidelity:
  - the degree to which the results produced by an application match the highest quality results that would be produced given ample computational resources

- Two reasons why a mobile computer operating alone may choose to reduce application fidelity:
  - to **achieve acceptable response times** for interactive applications
  - to **extend the mobile computer's battery** lifetime

- For demanding applications:
  - fidelity reduction may sometimes be absolutely necessary in order to get the application to run on the mobile device at all
  - e.g. a fixed computer may be able to execute a intense computation that a mobile computer is simply incapable of doing locally due to insufficient memory or storage.

- Offloading a computation or data query to a cluster in the cloud provides access to orders of magnitude more resource than a mobile computer has available locally

Costs of Using Remote Infrastructure (1/3)

- Offloading application from the mobile computer may degrade performance:
  - the mobile computer must **ship the inputs** to that computation to the remote infrastructure over a wireless network
  - after the computation completes, the mobile computer must **retrieve the results** of the computation
- Unless the computation is asynchronous and not on the critical path of the application, **performance is improved only if** the:
  - time saved by performing the computation remotely is greater than
  - the time spent communicating inputs and outputs at the start and end of the computation
- When
  - **network latency** between the mobile and remote computers is **high**,
  - **bandwidth is low**, or
  - the **amount of data shipped** per unit of computation is **large**
- it may be **quicker to perform the computation on the mobile computer** than it would be to perform the computation remotely

Costs of Using Remote Infrastructure (3/3)

Factor	Favorable for remote execution	Favorable for local execution
Network bandwidth	High	Low
Network latency	Low	High
Disparity between compute resources	High	Low
Granularity of computation	Large	Small
Parallelism in computation	High	Low
Memory/storage requirements	High	Low
Size of inputs/outputs	Small	Large
Load on remote computers	Low	High
Wireless network power	Low	High
Sensitivity of activity	N/A	Sensitive

Candidate Partitions

Where can an app be split?

Candidate Partitions

- How to pick a method for determining candidate partitions for applications?

- Theoretically:
  - the **number of possible application partitions is very large** since partitioning could employ granularities as fine as single instructions

- From a practical standpoint:
  - **very fine-grained partitions are ineffective** because substantial computation is needed to offset the performance and energy costs of using the network
  - further, the **performance overhead of the partitioning decision must be small** since it is executed dynamically; this limits the number of possible partitions that can be considered

- Thus, cyberforaging systems typically:
  - first **enumerate** a small number of possible partitions,
  - which we call the **candidate** partitions, and then
  - dynamically **select** the candidate partition that best achieves the system goal

Candidate Partitions - granularity

- There is considerable difference of opinion about how best to enumerate candidate partitions

Cyber Foraging System	Partition Granularity	Programmer Effort
RPF	Whole application	None
Spectra	Programmer-specified	Identify components
Chroma	Programmer-specified	Identify components
MAUI	Method	Annotate remotable methods
CloneCloud	Method	None
Odessa	Sprout component	None, but must use Sprout *

*Sprout is a stream processing system. Sprout models applications as a data flow graph in which the vertices are computational components called stages; Odessa dynamically decides where stages should be placed and when to employ parallel computation.

Candidate Partitions – programmer effort

- **Ask a programmer** to specify the ways of partitioning an application (Chroma):
  - this approach is based on the belief that for every application, the number of useful ways of splitting the application for remote execution is small

- The **number of candidate partitions is small**, and the **granularity of partitioning is typically large** application components:
  - e.g., in a natural language translator: four components for partitioning (three translation engines and a language modeler) and seven candidate partitions among those four components
  - a language (e.g. called Vivendi) allows **developers to specify a compact static partition** of all meaningful partitions of an applications
    - an application developer specifies code components called **remoteops** that may benefit from remote execution given the right circumstances
    - the developer also specifies **tactics**, each of which is a different way of combining remote procedure calls to produce a remoteop result
    - at runtime, the **Chroma system selects** one of the specified tactics

- Many other systems follow this approach:
  - e.g., Spectra **asks the developer** to specify candidate partitions explicitly
  - RPF uses a more extreme version of this approach in which it considers only **two candidate partitions**: one in which the core computation is performed on the mobile computer and one in which the computation is performed remotely

Candidate Partitions – no programmer effort

- The other extreme is to choose candidate partitions at method granularity without requiring **any programmer assistance**:
  - this approach takes advantage of modern language runtimes to **identify application components** that can be remotely executed and the data on which those components operate

- CloneCloud offers the best example of this approach:
  - it **identifies candidate partitions through static analysis** of code compiled to run in Android's Dalvik virtual machine
  - it **prunes the set of candidate partitions** by considering only those partitions that start and end at VM-layer method boundaries
  - it imposes **three additional restrictions**:
    - **methods that access platform-specific features** (such as the GPS device on a mobile phone) must execute on the platform that has those features
    - **methods that access the same native state** (e.g., a temporary file) must execute on the same computer
    - **nested migration is disallowed** — if a method is migrated from the mobile computer to remote infrastructure, the methods it invokes must also execute remotely

  - while one can imagine removing each of these restrictions (e.g., by supporting remote service invocation or migration of native state):
    - the **restrictions support the common case** while substantially **reducing the set of candidate partitions** that CloneCloud needs to consider

Candidate Partitions – method granularity

- Other systems use a **hybrid** of these two approaches

- E.g. MAUI:
  - it also considers **candidate partitions specified at method granularity**
  - however, it **asks** developers to specify which methods it should consider executing remotely by annotating those methods with the custom attribute feature of the Microsoft .NET Common Language Runtime

- This requires that the developer perform some of the checks that are done automatically by CloneCloud (for example):
  - **verifying that the method does not access functionality specific** to the mobile platform,
  - it also **allows the developer to pass along domain-specific knowledge** (for instance, that it seldom is a good idea to execute user interface methods remotely)

Candidate Partitions – component granularity

- Language runtimes represent only one way for a cyber foraging system to automatically extract application components

- If an **application has already been divided into components** to use distributed systems middleware:
  - the cyberforaging system can **leverage the same component structure** to identify candidate partitions

- E.g., Odessa targets applications that have already been modified to use the Sprout distributed stream processing system:
  - Sprout models applications as a **data flow graph** in which the vertices are computational components called stages
  - Odessa dynamically decides **where stages should be placed** and **when to employ parallel computation**

Partitioning Metrics

How do we know we picked the right partition?

Partitioning Goals - maximize one metric

- There is some metric to be maximized:
  - **there is no consensus among cyber foraging systems about which of these metrics to employ**
  - most systems only consider a subset of them when deciding where to place application components

Cyber Foraging System	Goals	Method of Relating Goals
RPF [Rudenko et al., 1998]	Energy	N/A
Spectra [Flinn et al., 2002]	Execution time, energy, and fidelity	Utility function and goal-directed adaptation
Chroma [Balan et al., 2003]	Execution time and fidelity	Utility function
MAUI [Cuervo et al., 2010]	Execution time and energy	Optimize energy subject to execution time constraint
CloneCloud [Chun et al., 2011]	Execution time or energy	N/A
Odessa [Ra et al., 2011]	Makespan and throughput	Only execute remotely if both goals improve

Partitioning Goals – relating metrics

- One method is to:
  - **focus on one metric that must respect some constraint**
  - e.g. minimize energy usage on the mobile computer subject to the constraint that performance is no worse than 105% of the performance that would be achieved by executing the activity entirely on the mobile device

- Another method is to:
  - execute functionality **only if all performance metrics indicate that remote execution is favorable**
  - e.g. executes application components on remote infrastructure only if it expects to improve both throughput and time taken to finish all processing as a result

- A third possible method:
  - defines **conversion factors between metrics**
  - e.g. assign equal importance to fidelity and execution time

- Each of the above methods represents a **type of policy** for relating metrics expressed in **different currencies**:
  - none of the above take into account **user preferences** and variable goals

Partitioning Goals – user help

- A few methods attempt to solicit user input to help set the policy dynamically (e.g. Spectra):
  - it is based on the observation that the **relative importance of metrics may change** over time
  - it **asks its user** to **specify a target battery lifetime** for the mobile device at runtime
  - it uses a **feedback process** termed goal-directed adaptation to **dynamically adjust** the relative importance of energy in relation to fidelity and execution time
  - it is based on how well the **rate** at which the mobile computer is consuming power energy matches the target battery lifetime

- Thus:
  - when the mobile computer is projected to **run out of battery energy too soon**, e.g. Spectra **increases the relative importance of energy-efficiency**
  - if the mobile computer is projected to **have substantial energy left over**, e.g. Spectra **decreases the relative importance of energy efficiency.**

- A similar approach could potentially be used to address other budgeted resources for which a fixed maximum consumption is allowed over a time period:
  - e.g., **goal-directed adaptation** has also been used to ensure that a mobile phone does not exceed a **monthly cap on a cellular data plan**

Partitioning Goals – off-line

- Aura uses **off-line profiling** to construct utility functions:
  - it allows **users to express thresholds** for satisfaction and starvation for multiple metrics
  - it then **interpolates** among those specifications using sigmoid functions

- In general, these systems illustrate that there is an inherent tradeoff in soliciting user input

- More user participation:
  - it can **improve the quality** of the partitioning, but
  - the participation may **distract the user** from the task at hand and prove **too burdensome**

Selecting a Partition

- Given a set of candidate partitions:
  - a cyber foraging system **selects** the one that it believes will best achieve its goals

- This should be the partition that:
  - **maximizes** the system's chosen metric or utility function while satisfying any constraints

- This is necessarily imperfect because making it correctly requires the cyber foraging system to **predict the future**

- An accurate prediction requires the system to:
  - **estimate** not only the conditions of the mobile computing **environment** (for example, network bandwidth, latency, and server load), but also
  - **application behavior** (e.g. the amount of computation that will be performed by a component for a specific set of inputs)

- Typically, the cyber foraging system assumes that **the past will predict the future**

- It observes application behavior and/or the resources available to the mobile computer and bases future predictions on those observations

Resource Measurement and Estimation

How can we measure the cost of offloading computation?

Resource Measurement and Estimation

- Resource supply:
  - the resources available in the mobile computer's computational environment
- Resource demand:
  - the requirements of each candidate partition
- Resources that cyber foraging systems most often use for **supply-and-demand estimation**:
  - CPU, network, battery energy, file cache state, and memory
- Cyber foraging systems (e.g. Spectra and Chroma) measure the demand (usage) of CPU for an application:
  - reading these statistics through interfaces such as Linux's /proc file system
- The per-application nature of the statistics:
  - enables the cyber foraging system
  - to distinguish the usage of the target application from other **concurrent activities** on the computer
  - although some **conflicting effects** (such as lock and cache contention) are very difficult to remove

How fast are the CPUs involved?

CPU Measurement and Estimation – first challenge

- **First challenge - measuring CPU supply and demand**:
  - such measurements must be performed on both the **mobile computer** and the **remote server(s)** used in cyber foraging

- The **remote server** employed for cyber foraging **may change** depending on such factors:
  - e.g. the location of the mobile computer

- This means that the cyber foraging system must:
  - either **thoroughly measure behavior on all computers** before making an accurate decision, or
  - it must **account for differences in processing power** across computers

CPU Measurement and Estimation – second challenge

- **Second challenge - arises from the distributed nature of cyber foraging**

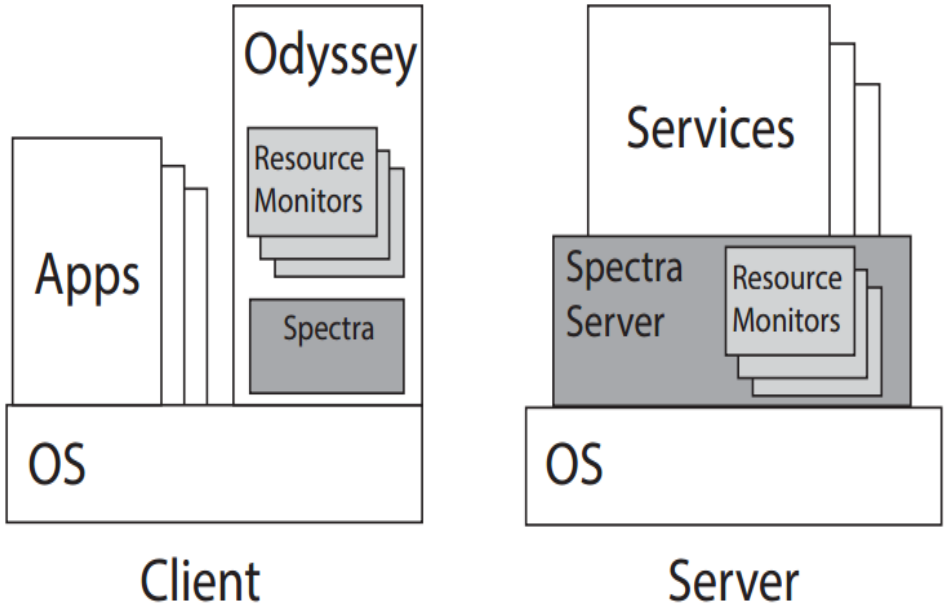
- The partitioning **decision is made on one computer** (often, but not always the mobile computer) **using as an input the CPU load of other computers**

- Thus, when a partitioning decision arises:
  - the cyber foraging system could potentially query all other computers that might participate in the computation for their current CPU load

- Unfortunately, this involves a network round trip:
  - if the **computation being considered is relatively small**, the extra time to perform the query of remote state could **add an unacceptable overhead** relative to the compute time
  - the opposite is also true: if the **computation is long-running**, then the relative **overhead of a network round-trip** may be insignificant

Spectra

- Spectra uses slightly stale estimates of remote state:
  - it runs **resource monitors** on both the mobile computer and remote servers
  - each monitor is responsible for a **different resource** such as CPU, network, etc
  - resource monitors on the **remote server periodically and asynchronously report the state** of the server to the mobile computer via remote procedure calls
  - to avoid a network round trip, Spectra uses the last received estimate in its partitioning decision, even though this **estimate may be several seconds old**



How fast is the network?

Network Measurement and Estimation (1/3)

- Cyber foraging systems measure network supply (i.e., the available bandwidth and latency) through a variety of strategies:
  - **active** measurement,
  - **passive** measurement, or
  - a combination of the two

- Passive measurements observe traffic sent over the network for other purposes:
  - they **do not inject additional traffic**
  - this **conserves both battery energy and cellular network usage**
  - however, **passive measurements may become stale** if the mobile computer does not send or receive data for a time

- Active measurements:
  - **inject traffic** into the network solely for the purpose of measuring the network performance
  - provide more recent data

Network Measurement and Estimation (2/3)

- Spectra uses passive measurements:
  - its network monitor **predicts bandwidth and latency**
  - its RPC package **logs the sizes and elapsed time of short exchanges and bulk transfers**
  - the **small RPCs are used to predict latency**, and the **larger RPCs are used to predict throughput**
  - it periodically **examines the total bandwidth available** to the mobile computer
  - it then **estimates the bandwidth** likely to be available for communicating with each server

- Passive measurement of network demand (bytes transmitted) is accomplished in a similar manner:
  - Spectra simply **queries the RPC package to determine the bytes required to transfer state** for each RPC performed

Network Measurement and Estimation 1/3

- MAUI:
  - uses **both active and passive** measurements
  - it employs the simple strategy of **sending 10 KB of data to the remote server and measuring the throughput** of the transfer directly to obtain an average throughput
  - this works well because the **10 KB size is representative of the typical transfers** to the MAUI server

- MAUI also uses passive estimation to refine its active measurements:
  - **every time that a method is offloaded**, MAUI obtains a **more recent observation of network quality** by measuring the duration of the transfer
  - **active measurements are only employed if no transfers have provided recent passive estimates**
  - if more than a minute passes without a transfer, MAUI sends 10 KB to the server and observes the result

What’s the impact of offloading on the battery?

Battery Measurement and Estimation

- The **supply of battery energy** is one of the **simplest resource values to measure and estimate**:
  - the supply of battery energy is just the amount of charge remaining in the battery
  - most mobile computers provide standard interfaces for querying this value

- The **demand for battery energy is one of the most difficult:**
  - it is the amount of batter energy consumed by a particular computation

- Cyber foraging systems typically use one of two methods to estimate this value:
  - direct measurement (e.g. Spectra and Chroma)
  - model-based approach

Battery Measurement and Estimation - direct

- Direct measurement (e.g. Spectra and Chroma)
  - **measure** the amount of energy in the mobile computer's battery **before an operation begins** and **after the operation completes**
  - the difference between the two values is the amount of energy required to perform the computation

- Direct measurement has two drawbacks:
  - if **more than one activity occurs simultaneously**, it is difficult to separate the energy cost of each activity
    - Spectra adjusts for this issue by discarding all energy demand measurements in which operations executed concurrently.
  - many mobile computers **do not provide fine-grained measurements** of battery capacity

- Report battery energy at **granularities as coarse as 1% of total battery capacity is insufficient** to measure brief computational events

Battery Measurement and Estimation - model

- Many cyber foraging systems use a **model-based approach** to estimate **energy demand**:
  - executes a series of **micro benchmarks on the mobile computer** in a **laboratory setting**
  - while an **external power measurement device measures how much energy** the computer expends executing those benchmarks
  - an **energy model is constructed** using the results of these experiments

- The model assigns specific energy costs to measurable activities performed by the mobile computer:
  - e.g. **executing** a given amount of computation, **sending** or **receiving** a byte of data over a wireless network, or **enabling** the screen backlight

Battery Measurement and Estimation – model

- Such models are naturally **specific** to a particular brand and model of a mobile device

- Given such an energy model a cyber foraging system can:
  - measure the occurrence of events such as computation and network transmission that occur during application execution,
  - then use the model to assign an energy costs to those events

- The end result is an estimate of the energy used to perform the activity

Battery Measurement and Estimation - model

- Drawback of using energy models:
  - they necessarily **do not capture all the potential factors** that may influence mobile computer energy usage

- E.g.:
  - a model might capture the **average energy consumed per CPU cycle of computation**

- However:
  - **different instructions consume different amounts of energy**, so the mix of instructions can affect the total energy consumed by an operation
  - the **amount of memory accesses per instruction** substantially changes the energy expenditure
  - finally, **processor power management** will substantially alter energy usage

Applications - examples

- RPF executes **compiling tasks**
- Spectra runs a **speech recognizer**, **document preparation tools**, and a **natural language translator**
- Chroma supports **face recognition**, **text-to-speech synthesis**, **image filtering**, **3D model viewing**, **speech recognition**, **music identification**, **natural language translation**, **lighting for 3D models**, and **creation of 3D scenes from 2D images**
- MAUI executes a **face recognizer**, a **video game**, and a **chess game**
- CloneCloud runs a **virus scanner**, an **image search application**, and **privacy-preserving targeted advertising**
- MARS targets three **computationally intensive applications**: **augmented reality pattern recognition**, **face recognition**, and an **augmented reality video game**
- Odessa supports **face recognition**, **object and pose recognition**, and **gesture recognition**

Applications – best tasks

- The tasks best suited for cyber foraging are **compute-intensive**

- Such tasks **benefit most from remote execution**, and they therefore **offer the best opportunity to recoup the costs of sending state over the wireless network**

- Cyber foraging is **not necessarily limited to batch tasks** like compiling and document processing:
  - most of the applications listed previously are interactive
  - however, **all have significant compute chunks** that **occur between interactions with the user**
  - the **length of such compute chunks must be substantial compared to the latency of the wireless network**, unless the computation can be performed asynchronously

Applications - multimedia

- Most of the recent applications that leverage cyber foraging involve **multimedia**:
  - either through **intense audio processing** (e.g., speech recognition and natural language translation), image processing (e.g., face recognition), or
  - **support for video** (e.g., games)

- Thus, the **trend to incorporate more multimedia** in everyday applications may **increase the opportunities to employ cyber foraging** in future applications

- Many of the applications listed above are especially relevant to mobile computer users:
  - applications such as **natural language translation** have clear applications for visitors traveling through a foreign country
  - **face recognition and music identification** are already used in popular mobile applications

- Thus, the future seems promising for cyber foraging:
  - **such applications demand capabilities that mobile computers may simply be unable to provide** with purely local resources while running with a limited battery capacity

Management

Management Issues

- Location and management of servers that host remote operations during cyber foraging
- We refer to a remote computer that may temporarily assist a mobile computer as a surrogate
- Where should surrogates be **located**?
- How should remote operations on a surrogate be **isolated**?
- How should surrogates acquire and manage the **state** associated with cyber foraging?

Location Decision Factors

- Factors that impact the location decision, including:
  - network latency and bandwidth, ease-of-management, cost, and privacy concerns

- Options for location:
  - deploying surrogates as close to the edge of the network as possible,
  - use of dedicated computers in a data centre, and
  - opportunistic use of computers in a data centre

- How any of these options may be best, depending on which of the factors listed above are considered the most important

Location Examples

- Network latency has a similar effect on energy usage:
  - the mobile computer expends power while waiting for the surrogate to return a result

- In the limit, it may not be feasible to offload any part of an application if a surrogate is located too far away

- Examples:
  - for a chess game, offloading methods to a surrogate improves performance only if the surrogate is located nearby — with a network round-trip time (RTT) of 25 ms or less

  - for face recognition and a video game, the performance difference between using a nearby (10 ms RTT) and a distant (220 ms RTT) surrogate can be as large as 50%

  - w.r.t. energy usage, the two applications (chess and the video game) consumed more energy when the mobile computer offloaded methods over 3G (incurring a 220 ms RTT) than if the method had not been offloaded,
  - whereas WiFi transmission, which led to lower round-trip times, enabled offloading to decrease the energy usage of a cell phone

Surrogates in the Cloud - benefits

- Ease of management:
  - much easier to replace a hardware component that has failed in a centralized data centre than it is to make a service call to a remote location to replace the same component
  - physical proximity may make it easier to troubleshoot software problems and repair mis-configurations
  - it is likely that network connectivity, power, and other external inputs to the surrogate are more reliable in a data centre than in a coffee shop or airport lounge

- Easier to provide physical security:
  - if surrogates are deployed in public locations, then additional measures, may be required to prevent tampering

Conclusions

- It's a hot topic and a research concept which has already practical use cases (mostly with static partitioning)

- Cyber foraging stands at the confluence of two strong trends in modern community:
  - every day more people access data and run computation from their mobile computers, and
  - data and computation is increasingly distributed among those devices and cloud data centers

- Cyber foraging is especially compelling in a post-PC world in which desktop and workstation computers are replaced by mobile and cloud devices

- The large variation in mobile computing environments argues that functionality should by dynamically, rather than statically, partitioned among mobile and cloud computers

- The limitations of locating computation in both cloud data centers (e.g., communication latency) and on mobile devices (e.g., limited resources):
  - create the need to insert a surrogate computational tier between these extremes that can mitigate both sets of limitations