

Programming Ubiquitous Things
2024/25 - Spring 2025 - UiO

Cyberforaging

Prof. Paulo Ferreira

paulofe@ifi.uio.no

UiO/IFI/PT

1

1



Contents

- Introduction (3-15):
  - motivation, benefits
- Partitioning (16-28):
  - what it is, goals, candidate partitions, selecting a partition, resource measurement and estimation, errors, applications
- Resource Measurement and Estimation (29-59):
  - CPU, network, battery, errors, replication, best applications
- Management (60-76):
  - surrogates location, isolation, common and application-state
- Data Staging (77-81):
  - definition, example

2

2



Introduction

3

3



Motivation – how to partition

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

- **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 mostly replaced personal computers, and
  - are now in turn being replaced by tablets and smartphones (in some cases)
- 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 servers and data stores provide **more computational resources than a mobile device**:
  - they are not limited by stringent size, weight, and battery energy constraints

4

4

Motivation – thin vs thick



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

- Current applications typically address this question in an ad-hoc fashion, either by operating in a:
  - **thin-client** mode, or in
  - **thick-client** mode
- Thin-client mode:
  - all computation/data access is performed at the fixed infrastructure
  - the **mobile device operates only as a terminal**
- Thick-client (or fat) mode:
  - the computation/data are hosted on the mobile device
  - wireless networks are used only to **fetch inputs not available locally**
- A handful of applications adopt a more balanced, but still **static**, **partitioning** of **functionality**

5

5

Motivation - static vs dynamic



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

- **Static** partitioning **fails** to account for the **variability** inherent in mobile and ubiquitous/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 and/or 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

6

6

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 ubiquitous/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
- This infrastructure may either be designated resources:
  - in the **cloud**, or
  - **nearby compute and storage resources** that are **dynamically identified** and **opportunistically exploited**
- As the conditions of the mobile **environment changes**:
  - cyber foraging systems **dynamically adjust the partitioning of computation and data** to provide the best user experience
- As long as sufficient infrastructure **resources can be harnessed** from the environment:
  - cyber foraging can provide mobile users with the **performance** they have come to expect from better-equipped computing platforms such as desktop computers

7

7

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**, more **storage**, less **battery** used, app **fidelity**
- There is a fundamental performance **gap between mobile and fixed computers**

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)

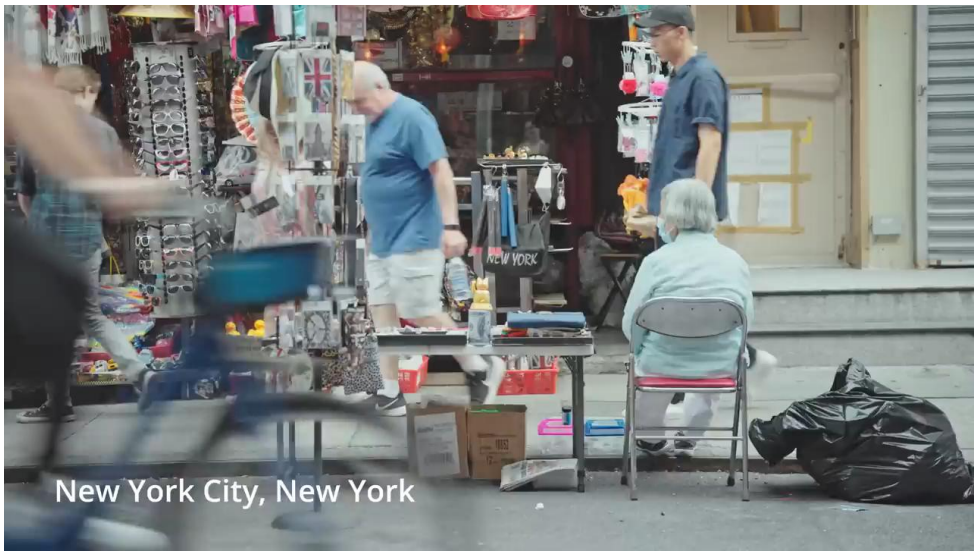
- Comparing the processing power of representative computer systems in five-year increments including the century crossing (1997–2012):
  - as a **rough estimate**, **processing power is given by the product of core count and clock speed**
  - although both server and mobile computers both show rapid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

8

8

Examples of Static Partitioning (1/3)

Introduction-https://www.youtube.com/watch?v=Z0gpN7V5zQs

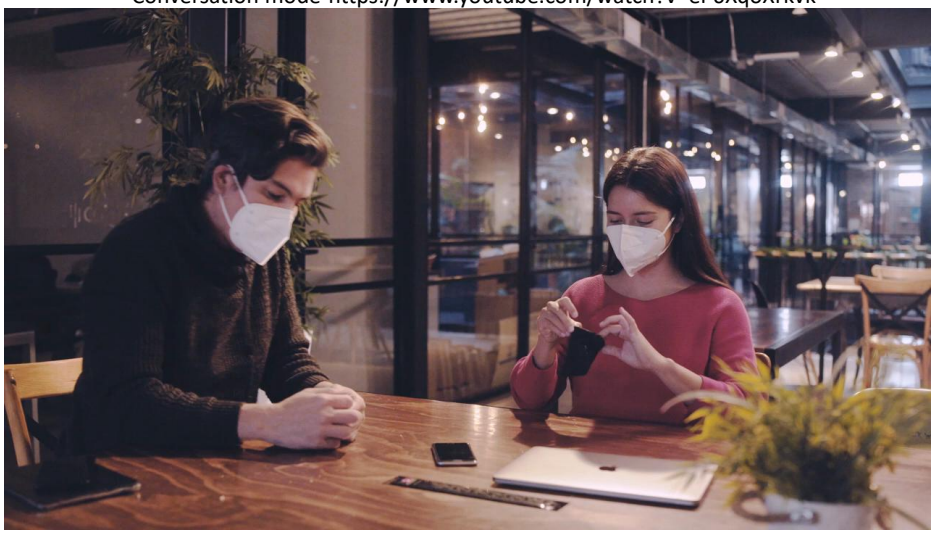


9

9

Examples of Static Partitioning (2/3)

Conversation mode-https://www.youtube.com/watch?v=eP6XqoXrkvk

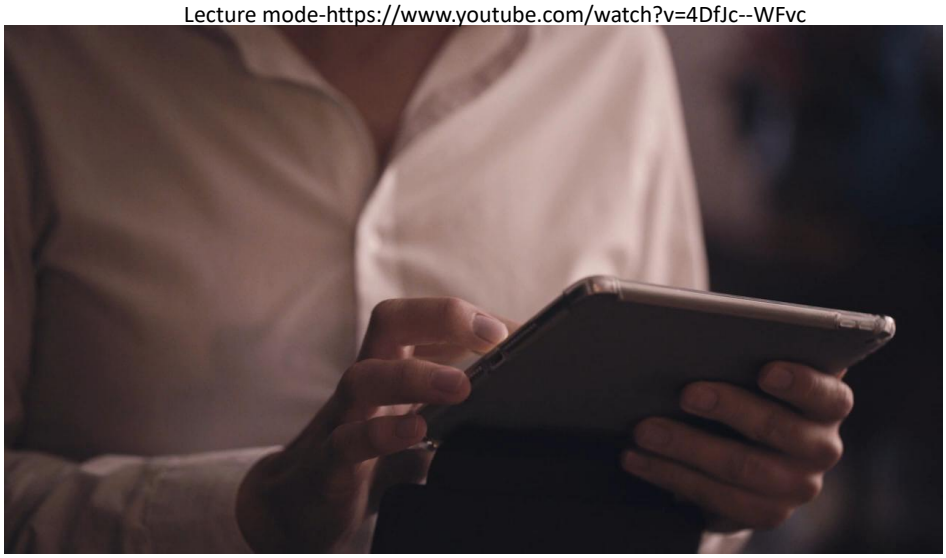


10

10

Examples of Static Partitioning (3/3)

Lecture mode-https://www.youtube.com/watch?v=4DfJc--WFvc



11

11

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**

12

12

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

13

13

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 an 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 resources than a mobile computer has available locally
- There are two choices:
  - **execute** the application **locally** with **reduced fidelity**, or
  - leverage **fixed infrastructure** to provide **full fidelity**

14

14

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**
- It may be quicker to perform the computation on the mobile computer than it would be to perform the computation remotely 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**

15

15

Costs of Using Remote Infrastructure (2/3)



- For **energy savings** the situation is similar:
  - the mobile computer may be able to use aggressive **hardware power-saving modes** while **application activity is performed remotely** (saving energy),
  - it must use a **power-hungry wireless network** to **initiate** the remote activity and **receive** the results
  - it may require more **energy** to **offload activity** to a remote computer than it would require to do the same activity locally
- Offloading application activity to remote infrastructure introduces **security and privacy** concerns:
  - e.g., an **attacker** with physical access to the infrastructure may cause an application to **return incorrect results**, or
  - an **eavesdropper** may monitor the remote computation to **discover private data**

16

16

Costs of Using Remote Infrastructure (3/3)



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

17

17

Cyberforaging Complexity



- The **partitioning decision is complex**:
  - a cyber foraging system must **balance numerous goals** (e.g., performance, energy efficiency, fidelity, privacy, and security),
  - it must also evaluate a **large number of factors** (including those shown in the last slide)
- All of the factors listed above may need to be **evaluated dynamically**:
  - **network** conditions and **load** may **change rapidly** in mobile environments
  - factors such as the **granularity of computation** and the **sensitivity of data** may depend upon specific inputs chosen by the user at runtime
- There is simply no **"one-size-fits-all"** choice for all of these factors
- It is **infeasible** to expect that a static partitioning decision can yield optimal or even good results in many environments in which a mobile computer may operate
- Achieving **good results requires** a **dynamic decision**

18

18



Partitioning

19



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
  - **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
  - what **characteristics** make applications particularly well suited or poorly suited for cyber foraging

20

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

- **No system currently considers security or privacy** explicitly when making dynamic partitioning decisions:
  - it is hard to **determine which data is sensitive** and which is not **without specific user annotations**
  - the **negative consequences** of making incorrect security and privacy decisions **can be large**
  - this means that all **security concerns** must currently be addressed **statically**, perhaps **limiting the set of applications** that can benefit from cyber foraging

21

21

Partitioning Goals – maximize several metrics

- Attempting to **optimize more than one metric** faces an additional challenge:
  - there is **no common currency** for these metrics
  - e.g., energy expenditure is expressed in **Joules**, performance in **seconds**, and fidelity is an **abstract** notion of application quality
- When a particular partitioning **optimizes all metrics**:
  - that partitioning is the obvious choice
- When no such partitioning exists:
  - the cyber foraging system must **sacrifice one metric** to improve another

22

22

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 relevant metrics indicate that remote execution is favorable**
  - e.g., **executes** application components **on remote infrastructure** only if it expects to **improve both throughput and makespan** (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

23

23

Candidate Partitions

- After choosing metrics and constraints that express the goals for partitioning:
  - must **decide** on 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, cyber foraging 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

24

24

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.

25

25

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: components for partitioning (translation engines and a language modeler) and candidate partitions among those components
  - a language (e.g., Vivendi) allows **developers to specify a compact static partition** of all meaningful partitions of applications
    - an application developer specifies code components called **remoteops** that may benefit from remote execution given the right circumstances
- Many other systems follow this approach:
  - e.g., Spectra
  - **Spectra asks the developer** to specify candidate partitions explicitly (as in Chroma)

26

26

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

27

27

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
- While 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)

28

28

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**
- Although not targeted specifically at mobile computing, Coign uses a similar strategy:
  - by targeting applications built using Microsoft's Common Object Model (COM), Coign can learn of partition boundaries and distribute an application without any knowledge of the application source code

29

29

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** (e.g., 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

30

30

Selecting a Partition - estimation

- Strategies employed by different cyber foraging systems to choose among candidate partitions:
  - the second column shows the method used to identify the best candidate
  - the third column summarizes the resources measured to make that identification

Cyber Foraging System	Cost/Benefit Estimation	Resources Measured
RPF	Direct observation	None
Spectra	History-based prediction	CPU, network, energy, and cache state
Chroma	History-based prediction	CPU, memory, network, energy, and cache state
MAUI	Supply and demand	CPU, network, and energy
CloneCloud	Supply and demand	Network
Odessa	Bottleneck identification	Network and remote parallelism

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

31

31

Resource Measurement and Estimation

32

32

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, 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

33

33

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

34

34

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

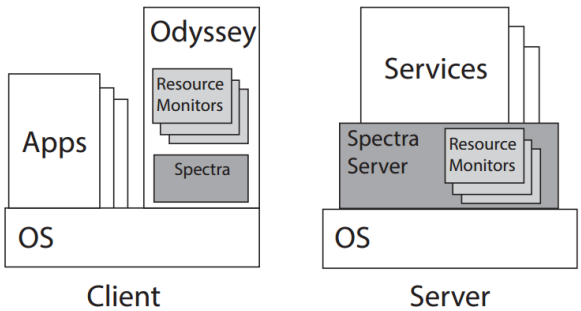
35

35

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**



36

36



Resource Measurement and Estimation – parallelism, network, and battery

37

CPU Measurement and Estimation - parallelism



- Most current cyber foraging systems do not yet adequately account for two modern trends in computer architecture:
  - first, **variable-speed processors** are pervasively used in mobile computers
    - if the required computation is not expected to require the full processing power of the computer, the processor is placed in a low-speed state in which it uses less energy to perform the computation
  - second, **parallelism** is ubiquitous, especially in server design, where processors employ multiple cores
- For computation that can be parallelized, e.g., using multiple threads:
  - the amount of **available cores** on a target computer is an important factor in determining how fast the computation will complete

38

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

39

39

Network Measurement and Estimation (2/3)



- Spectra uses passive measurements for **network supply**:
  - 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

40

40

Network Measurement and Estimation (3/3)



- 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 uses passive and active estimation:
  - **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

41

41

Network Measurement and Estimation – challenges (1/4)



- Current hardware trends are making network estimation more challenging:
  - use of **power-saving modes** for network interfaces
- Network interface is one of the dominant sources of energy consumption for small, mobile computers:
  - such computers place the wireless interface in a **low power mode** when it is not sending or receiving packets
- 802.11 standard defines:
  - power saving mode (PSM) **disables the interface** for short periods (typically 100 ms)
  - since the interface cannot receive packets when it is disabled, the 802.11 base station **queues data until the interface reactivates**

42

42

Network Measurement and Estimation – challenges (2/4)



- PSM can **cause substantial performance degradation** for short request-response traffic:
  - the **mobile computer** sends the request and waits for a response
  - the **network interface enters power-saving mode** shortly after the request is sent since it sees no network traffic
  - the **response arrive a short time later** but the **base station must delay sending it** for almost 100 ms **until the network interface of the mobile computer reactivates**
  - this **can lead to more than an order of magnitude slowdown** for short RPCs
- This effect is a substantial problem for cyber foraging systems:
  - especially those that try to **offload work as small as a single method**
  - even for **larger transfers**, the **degradation of disabling the interface can be seen due to TCP slow-start**

Network Measurement and Estimation – challenges (3/4)



- Mobile computers often have more than one connectivity option:
  - a cell phone might choose to send data over a **WiFi network** (if it is located near a hot spot) or
  - a **cellular network** (if it has a data plan)
- Networks may have **different characteristics**:
  - e.g., higher bandwidth, lower latency, or a lower loss rate, and
  - the **best network to use may change over time or as a user moves**
- Further, if cellular data plans enforce limits for the amount of data that may be sent within a time period:
  - **opportunistic offload of communication from 3G/4G cellular to WiFi networks**

Network Measurement and Estimation – challenges (4/4)



- The performance of offloaded computation might benefit from applying strategies that:
  - either **stripe data over multiple networks**, or
  - route traffic over the **most appropriate** of the available wireless network
- Thus, cyber foraging systems may need to **simultaneously consider conditions for more than one wireless network**
- However, **predicting the bandwidth and latency that can be achieved when using more than one network is more complicated** than predicting those values for a single network

45

45

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 battery 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**

46

46

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

47

47

Battery Measurement and Estimation – model (1/4)



- 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

48

48

Battery Measurement and Estimation – model (2/4)



- 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 (3/4)



- 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

Battery Measurement and Estimation – model (4/4)



- Thus, in developing any energy model, there is a **trade-off between complexity and accuracy**:
  - it is possible to **capture any one of the above effects** given sufficient micro benchmarks, offline analysis, and online accounting
  - however, capturing all possible effects for CPU, network, storage, display, and other components is a very **daunting task**
  - most cyber foraging systems compromise by using **simpler models** and accepting that demand measurements will include a **moderate amount of noise**

51

51



Errors During Partitioning

55

55

Errors During Partitioning (1/2)



- Most cyber foraging systems developed to date:
  - **do not explicitly consider such errors** or
  - have only **rudimentary strategies** to cope with their occurrence
- We:
  - elaborate on **some of the sources of inaccuracy and failure during partitioning**, and
  - speculate about some **possible strategies** that could cope with these errors
- One or more of the components of a cyber foraging system may fail during operation:
  - the objective of most cyber foraging systems is to **provide behavior equivalent to the behavior that would be seen if the application ran entirely on a mobile computer**, albeit with better performance
  - if the **mobile computer itself were to fail** (for example, after running out of battery power), an application running on that computer would lose all non-persistent state
  - it is usually trivial for a cyber foraging system to emulate this behavior by **canceling any remote operations** when a mobile computer fails

56

56

Errors During Partitioning (2/2)



- **Failure of a remote component** such as a wireless network or a remote server requires **more effort to mask**:
  - if the cyber foraging system has **offloaded computation to a remote computer that fails**, the state of the computation is lost
  - even if the failure is due to a **network partition**, waiting for a network connection to be reestablished may introduce unacceptable delay for interactive applications
- The cyber foraging systems that explicitly consider such failures often use **simple timeout mechanisms to detect and recover from them**

57

57

Errors During Partitioning – timeout (1/2)



- The difficulty in employing timeout-based strategies lies in **choosing the correct timeout value**
- If the **timeout is too short**:
  - the cyber foraging system will **abandon some remote computations** that would **shortly complete** after a transient failure
  - such **transient failures** are especially common in mobile computing due to the vagaries of wireless networking
- If the **timeout is too long**:
  - the cyber foraging system will **wait unnecessarily** for a remote operation to complete before initiating a recovery action
  - this will **increase interactive delay**
- There is **no perfect timeout value**:
  - some failures will be permanent,
  - some transient failures will require an excessively long time to recover, and it may be possible to recover from other transient failures almost immediately

58

58

Errors During Partitioning – timeout (2/2)



- **Changes in the mobile environment** can produce concerns related to those introduced by the possibility of **remote component failure**
- Even if the cyber foraging system can **accurately predict the current resource availability** at the time a component begins executing:
  - those **conditions can change dramatically** for the worse during the execution of the application component
  - e.g., the **load on a shared remote server** may increase substantially, or the **bandwidth available** over a wireless network may decrease rapidly
- The **same timeout-based strategies** employed to survive remote component failure can also address such scenarios:
  - the timeout for a remote operation will expire because the operation cannot complete on time due to increased server load or reduced network quality, and
  - the mobile computer will restart the computation after reevaluating the partitioning decision
- **Redundancy is a potential solution** to the above concerns that has **not yet been fully explored** in the context of cyber foraging:
  - **running the same application** component **simultaneously** on the mobile computer and on a remote server

59

59

Replication (1/2)



- Replication may be **active** or **passive**
- Active replication:
  - **redundant replicas are started concurrently**
- Passive replication:
  - **additional replica is started on demand**
  - it seems an attractive replacement for timeout-based cancel-and-restart of an unresponsive computation
- If the cyber foraging system **loses contact with a remote server**:
  - it **could start an additional computation locally or on another server**,
  - **use the first response it receives** (whether it comes from re-establishing contact with the lost server or from completing the newly started computation), and
  - **canceling the computation that has not yet responded**

60

60

Replication (2/2)



- Systems that replicate computation must carefully handle operations that require **exactly once semantics:**
  - e.g., sending network messages to external parties
- During failure-free operation:
  - one replica is typically designated the **primary** and is **responsible for sending such messages**
  - **messages from other replicas are suppressed**
  - when a **failure occurs**, **message logging techniques can ensure that only one instance of an exactly once operation is performed**
  - alternatively, **protocols can be designed so that duplicate messages sent during failover are dropped** (for example, a sequence number can be added to all messages)

61

61

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**

62

62

Which ones are the best tasks ?

63

63

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 recover from the costs of sending state over the wireless network**
- Cyber foraging is **not necessarily limited to batch tasks** like compiling:
  - 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

64

64

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

65

65



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?
- What are the steps required to **discover and start using a surrogate**?

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 center**, and
  - **opportunistic** use of computers in a data center
- How any of these options may be best, depending on which of the factors listed above are considered the most important

68

68

Surrogates near Mobile Computers



- Two possible extremes:
  - surrogates placed as **near to mobile computers** as possible
  - mobile devices connected to servers in the **cloud**
- Must always consider that:
  - **network latency** degrades the user experience,
  - leading cloud developers to **replicate services** across geographically distributed data centers or **cache** content closer to clients
- Locating **surrogates closer to mobile computers** has substantial performance benefits:
  - especially for highly interactive applications
- As the distance (in terms of network latency) between the mobile computer and the surrogate increases:
  - larger and larger computation chunks are required to recover the time lost due to network communication
- **Network latency** has a similar effect on **energy usage**:
  - the mobile computer expends power while waiting for the surrogate to return a result
- It may not be feasible to offload any part of an application if a surrogate is located too far away

69

69

Location Examples



- 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, 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 smartphone**

70

70

Surrogates in the Cloud - benefits



- **Ease of management**:
  - much easier to replace a hardware component that has failed in a centralized data center 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 misconfigurations
  - it is likely that network connectivity, power, and other external inputs to the surrogate are more reliable in a data center 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

71

71

Location Co-location



- Improve the performance of offloaded computation when that **computation interacts with other cloud components or accesses data stored in the cloud**:
  - if such a computation is hosted near the mobile computer, then each communication with cloud resources may incur high latency and/or limited throughput, especially if the last-hop link connecting the surrogate to the cloud is poor
  - this property can be viewed through the lens of a partitioning decision
  - if an application component interacts with data and other applications hosted in a data center more than it interacts with the mobile computer, then it will usually make more sense to execute that component on a surrogate located in that data center rather than on a surrogate located near the mobile computer

72

72

Cloudlet



- A **cloudlet** is defined to be:
  - a **trusted**, **resource-rich** computer or cluster of computers that's **well-connected to the Internet** and available for use by **nearby mobile devices**
- Cloudlet:
  - it can provide **low-latency**, **one-hop**, **high-bandwidth wireless connectivity** to mobile devices through physical proximity
- This vision requires that:
  - cloudlets be physically dispersed at a **scale much greater** than that used by current Internet service deployments
- Much like current WiFi access points, **cloudlets might be deployed**:
  - coffee shops, doctors' offices, and other places of business to provide reasonable coverage for mobile users

73

73

Isolation of Remote Operations



- The choice of how to **isolate operations running on a surrogate** is orthogonal to the choice of where to locate surrogates
- Cyber foraging systems uphold the following **result-equivalence property**:
  - the observable results of an operation that executes remotely on a surrogate should be indistinguishable from results that could have been produced by the same operation if it had executed on the mobile computer
- This is typically accomplished by **isolating the operation** within an execution sandbox on the surrogate

74

74

Sandbox Functions



- It provides a set of dependencies such as libraries, operating systems, and services:
  - typically, these dependencies are the same as those on the mobile computer
  - thus, an operation that invokes a dependency on the surrogate can expect a result that could have been returned by invoking the same dependency on the mobile computer
- The execution sandbox typically **limits and/or mediates the interactions** with dependencies external to the sandbox:
  - e.g., a sandbox might prevent an operation hosted on a surrogate from invoking services provided by the surrogate's operating system because that operating system may not be the same as the operating system running on the mobile computer (and hence may return results impossible to generate on the mobile computer)
- The **execution sandbox** may suppress duplicate messages and other external effects for cyber foraging systems that replicate application components on multiple surrogates

75

75

Sandbox Functions - goals



- There are **many possible methods for providing execution sandboxes** (this is the unit of isolation)
- This decision is one of the more substantive design choices in deploying surrogates for a cyber foraging system
- Goals:
  - **safety**, or the degree to which result-equivalence is guaranteed
  - **flexibility** is desirable to support remote execution of the greatest number of possible operations
  - **transparency**, the execution environment on the surrogate and on the mobile computer would be the same, so the same binaries can be used to execute an operation in either location
  - **minimize the performance overhead** of isolation fast start-up, is strongly correlated with the goal of reducing overhead

76

76

Sandbox Functions – VM and JVM



- Early cyber foraging systems relied on the **process-level isolation** provided by the operating system
- More modern cyber foraging systems have leveraged **virtualization**:
  - either through emulation of a specific hardware architecture (such as the x86 ISA) or
  - through application virtualization provided by managed runtimes for languages such as Java and C#
- The abstraction chosen for isolation also affects **programmer flexibility and transparency**
- **Java Virtual Machines** and other managed run-times are designed to be portable across many different types of computer hardware
- In contrast, **process-level operation** often restricts the operating system type and version that a surrogate must run
- The **isolation choice** also impacts the amount of state associated with an operation, which may in turn affect performance overhead and start-up costs

77

77

Sandbox Functions - summary

- Process isolation
- Application VM
- Hardware VM

	Process Isolation	Application VM	Hardware VM
Representative Systems	RPF Spectra Chroma	MAUI CloneCloud Scavenger	Slingshot Cloudlets
Safety	Poor	Moderate	Excellent
Flexibility	Limited by surrogate environment	Limited to specific language	Very flexible
Transparency	Poor	Excellent	Excellent
Overhead	Minimal	Moderate	Moderate
Start-up Time	Minimal	Moderate	Substantial

- Containerization (e.g., Docker) is also a possible approach

78

78

Managing State – common state

- Operations that execute on surrogates require **two types of state**:
  - common state
  - application state
- **Common state**:
  - the host operating system and software libraries installed on the surrogate, a hardware virtual machine monitor, or application VM software such as the Dalvik VM or the Common Language Runtime
  - it is the lowest common denominator for remote execution:
    - it represents the set of functionality that a particular cyber foraging system can assume about a surrogate to which it sends remote operations

79

79

Managing State – application state



- **Application state** is application specific:
  - data, source code, bytecode, or binary that executes the remote operation, files, etc.
  - it is any data required for a remote operation on a surrogate that is not part of that surrogate's common state
- The common state is **persistent and read-only**
- The application-specific state is **mutable**
- Application-specific state:
  - may persist on a surrogate only for a single remote operation
  - may allow application-specific state to remain on the surrogate for the duration of a single connection
  - may store application-specific state between connections, to reduce start up costs if the same mobile computer connects to the surrogate again in the future

80

80

Managing State – soft vs hard state



- The decision about **how long application-specific state may persist** on a surrogate is often driven by the size of the state
- Most cyber foraging systems require that all such **application-specific state be soft state**:
  - the surrogate does not guarantee that it will reliably store a copy of the stat
  - the state may be destroyed due to a **surrogate fault**, or the surrogate may simply decide to **discard the state** if it is running low on resources
- The **prohibition against storing hard state on most surrogates** implicitly derives from:
  - the transitory nature of the relationship between mobile computers and surrogates
- If there exists a **dedicated surrogate for the user in the cloud**:
  - it is reasonable to assume that appropriate data replication and backup schemes are available to safely store hard state
- If **surrogates are discovered and used opportunistically**:
  - mobile computers may not trust the reliability of third parties managing those surrogates
  - it is safer therefore to ensure that no state is permanently kept if such a surrogate fails

81

81



Data Staging

Data Staging – the idea



- By **temporarily storing** (on the surrogate) data sent from the PC:
  - a surrogate can enable the mobile computer to transfer data over a high bandwidth, one-hop wireless link rather than use a wide-area connection
  - this, can reduce the energy usage of the mobile computer as well as cellular bandwidth consumption
  - this can improve availability and performance
- **Speculatively prefetch** data items to a surrogate:
  - data may later be retrieved by a mobile computer when it passes nearby
  - to be effective, such systems much predict the future (possibly with help from user):
    - e.g., the data that the mobile computer will require, the path of the mobile computer's user, and the bandwidth available along that path
- Successful staging requires **predicting the future**:
  - the surrogate must anticipate the arrival of the mobile computer and the set of data that it will request

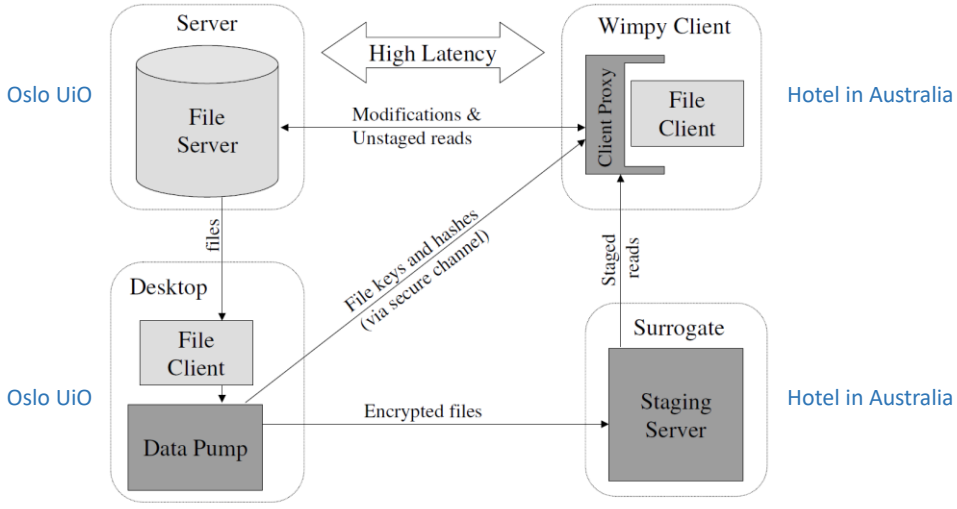
Data Staging – benefits

- Data staging:
  - offers **greater throughput**
  - staging data transfers from the cloud to the mobile computer can have substantial **energy benefits** as compared to unstaged transfers over WiFi and cellular networks
  - can also allow mobile computers to be more **cost efficient** (e.g., wifi vs cellular)
- Transmitting data as fast as possible is more energy efficient than sending data at slower speeds:
  - the best data transmission strategy is simply to **transmit data as fast as possible**, so that the device can return to a low-power mode as soon as possible

85

85

A Data Staging Example



86

86

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