CloneCloud: Elastic Execution between Mobile Device and Cloud

Piotr Karasiński
pk291580@gmail.com

31/11/2011

List of contents

List of contents

List of contents

List of contents

List of contents

List of contents

Actuall trends

- mobile devices are becomeing current computing wave

Actuall trends

- mobile devices are becomeing current computing wave
- mobile devices are competing heavily with desktops and laptops for market and popularity

Actuall trends

- mobile devices are becomeing current computing wave
- mobile devices are competing heavily with desktops and laptops for market and popularity
- they offer even more functionality then laptops (GPS, 3G)

Actuall trends

- mobile devices are becomeing current computing wave
- mobile devices are competing heavily with desktops and laptops for market and popularity
- they offer even more functionality then laptops (GPS, 3G)
- application for mobile devices need more and more resources (computation, storage)

Actuall trends

- mobile devices are becomeing current computing wave
- mobile devices are competing heavily with desktops and laptops for market and popularity
- they offer even more functionality then laptops (GPS, 3G)
- application for mobile devices need more and more resources (computation, storage)
- mobile devices run on extremly limited supply of energy

The question is:

Question

How to solve problem of limited resources of mobile devices and rising demands for computing, storage, etc.?

The question is:

Question

How to solve problem of limited resources of mobile devices and rising demands for computing, storage, etc.?

Obvious answer

Let's do it in clouds.

The question is:

Question

How to do it?

The question is:

Question

How to do it?

Answer

A lot of diffrents approaches, each of them have some strong and weak points.

The common apporach:

Split in traditional client-server paradigm, pushing most computation to the remote server.

The common apporach:

Split in traditional client-server paradigm, pushing most computation to the remote server.

The biggest drawback?

Seperate server to each application.

The common apporach:

Split in traditional client-server paradigm, pushing most computation to the remote server.

The biggest drawback?

Seperate server to each application.

Examples:

- Online navigations.
- Online dictionaries.

Goals

Main goal

"... a flexible architecture for the seamless use of ambient computation to augment mobile device applications, making them fast and energy-efficient."

Goals

Main goal

"... a flexible architecture for the seamless use of ambient computation to augment mobile device applications, making them fast and energy-efficient."

Others aims:

Goals

Main goal

"... a flexible architecture for the seamless use of ambient computation to augment mobile device applications, making them fast and energy-efficient."

Others aims:

- allow such grained flexibility on what run where (on mobile or in cloud)

Goals

Main goal

"... a flexible architecture for the seamless use of ambient computation to augment mobile device applications, making them fast and energy-efficient."

Others aims:

- allow such grained flexibility on what run where (on mobile or in cloud)
- take programmer out of the business of application partitioning (make it automatic and seamless)

Schema

1. Clone an unmodified application executable.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.
4. Remaining functionality on the mobile device keep executing.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.
4. Remaining functionality on the mobile device keep executing.
5. Remaining functionality blocks if attemps to acces migrated state.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.
4. Remaining functionality on the mobile device keep executing.
5. Remaining functionality blocks if attemps to acces migrated state.
6. The migrated thread executes on the clone, possibly accesing native features of hosting platforms (fast CPU, hardware accelarators).

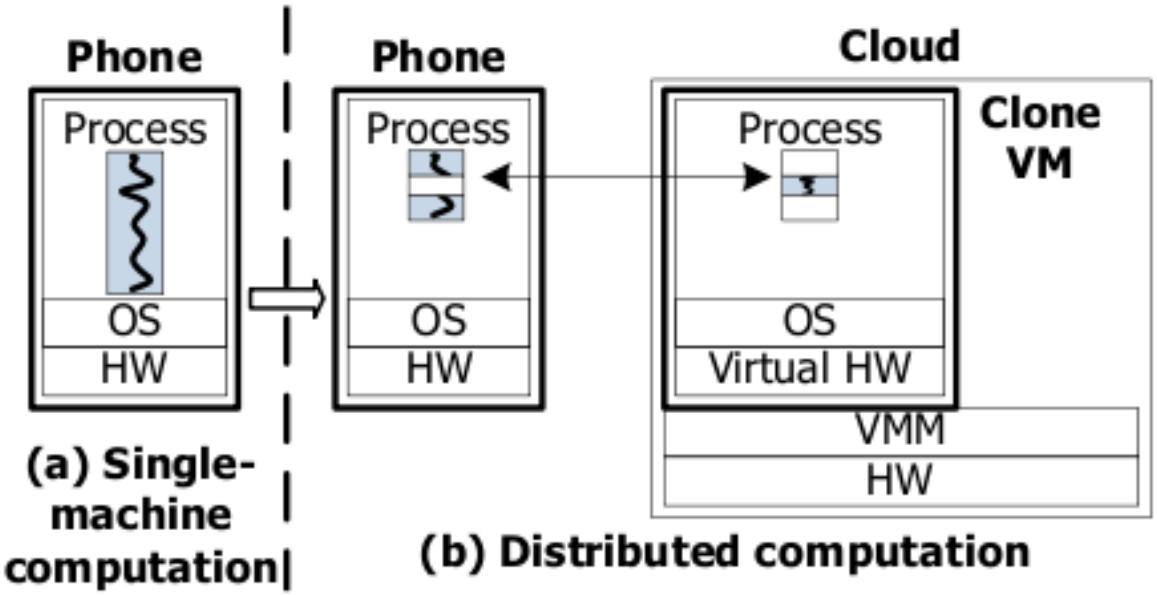
Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.
4. Remaining functionality on the mobile device keep executing.
5. Remaining functionality blocks if attemps to acces migrated state.
6. The migrated thread executes on the clone, possibly accesing native features of hosting platforms (fast CPU, hardware accelarators).
7. The remote thread returns back to the mobile device with remotely created state.

Schema

1. Clone an unmodified application executable.
2. The modified executable is running at mobile device.
3. At automatically chosen points individual threads migrate from mobile device to a device clone in a cloud.
4. Remaining functionality on the mobile device keep executing.
5. Remaining functionality blocks if attemps to acces migrated state.
6. The migrated thread executes on the clone, possibly accesing native features of hosting platforms (fast CPU, hardware accelarators).
7. The remote thread returns back to the mobile device with remotely created state.
8. Merge remote and local state back into original process.

Idea


Phone
Process
OS
HW
(a) Single-machine computation
Phone
Process
OS
HW
Cloud
Clone VM
Process
OS
Virtual HW
VMM
HW
(b) Distributed computation

Division

1. Choice where to migrate (made by partitioning component)

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer
   - Dynamic Profiler

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer
   - Dynamic Profiler
   - Optimization Solver

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer
   - Dynamic Profiler
   - Optimization Solver
2. Distributed Exectution

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer
   - Dynamic Profiler
   - Optimization Solver
2. Distributed Exectution
   - Suspend and Capture

Division

1. Choice where to migrate (made by partitioning component)
   - Static Analyzer
   - Dynamic Profiler
   - Optimization Solver
2. Distributed Exectution
   - Suspend and Capture
   - Resume and Merge

Overview

Partitioning mechanism decide which parts of an application's execution retain on the mobile device and which to migrate to the cloud.

Overview

Partitioning mechanism decide which parts of an application's execution retain on the mobile device and which to migrate to the cloud.

- offline

Overview

Partitioning mechanism decide which parts of an application's execution retain on the mobile device and which to migrate to the cloud.

- offline
- any application targeting the application VM platform may be partioned

Overview

Partitioning mechanism decide which parts of an application's execution retain on the mobile device and which to migrate to the cloud.

- offline
- any application targeting the application VM platform may be partioned
- the programer need no writing any annotioan or special idioms

Overview

Partitioning mechanism decide which parts of an application's execution retain on the mobile device and which to migrate to the cloud.

- offline
- any application targeting the application VM platform may be partioned
- the programer need no writing any annotioan or special idioms
- the partitioning mechanism may be run multiple times for diffrent execution conditions, resulting in a database of partitions

Static Analyzer

Static Analyzer

use static analysis to identify legal choices for placing migration and re-integration poits in the code. In principle, these points could be place anywhere in the code.

Restriction in CloneCloud

- restrict these points to be method entry (migration points) or method exit (re-integration points).

Restriction in CloneCloud

- restrict these points to be method entry (migration points) or method exit (re-integration points).
- allow migration only at boundaries of application method, not core system-library

Restriction in CloneCloud

- restrict these points to be method entry (migration points) or method exit (re-integration points).
- allow migration only at boundaries of application method, not core system-library
- allow migration at VM-layer method boundaries, not native method boundaries

Restriction in CloneCloud

- restrict these points to be method entry (migration points) or method exit (re-integration points).
- allow migration only at boundaries of application method, not core system-library
- allow migration at VM-layer method boundaries, not native method boundaries
- note, however, authors disallow migration while already in native execution, they allow migrated methods to invoke native ones

Property of any legal partition

1. Methods that acces specific features of a machine must be pinned to the machine

Property of any legal partition

1. Methods that acces specific features of a machine must be pinned to the machine
2. Methods that share native state must be collocated at the same machine.

Property of any legal partition

1. Methods that acces specific features of a machine must be pinned to the machine
2. Methods that share native state must be collocated at the same machine.
3. Prevent nested migration.

Dynamic Profiler

Dynamic Profiler

collects the data that will be used to construct a cost model for the application under different execution settings. The cost metric can vary, but this prototype uses execution time and energy consumed at the mobile device.

General mechanism

- the profiler is invoked on multiple executions of the application, each using a randomly chosen set of input data

General mechanism

- the profiler is invoked on multiple executions of the application, each using a randomly chosen set of input data
- each run executed once on the mobile device and once on the clone in the cloud

General mechanism

- the profiler is invoked on multiple executions of the application, each using a randomly chosen set of input data
- each run executed once on the mobile device and once on the clone in the cloud
- for each execution the output are profile trees T and T , from the mobile device and the clone, respectively

Profile tree

Profile tree

is a compact representation of an execution on a single platform.

- one node for each method invocation in the execution

Profile tree

Profile tree

is a compact representation of an execution on a single platform.

- one node for each method invocation in the execution
- rooted at the starting method invocation of the application (e.g. main)

Profile tree

Profile tree

is a compact representation of an execution on a single platform.

- one node for each method invocation in the execution
- rooted at the starting method invocation of the application (e.g. main)
- method calls in the execution are represented as edges from the node of the caller method invocation (parent) to the nodes of the callees (children)

Profile tree

Profile tree

is a compact representation of an execution on a single platform.

- one node for each method invocation in the execution
- rooted at the starting method invocation of the application (e.g. main)
- method calls in the execution are represented as edges from the node of the caller method invocation (parent) to the nodes of the callees (children)
- node is annotated with the cost of its particular invocation in the cost metric (execution time in our case)

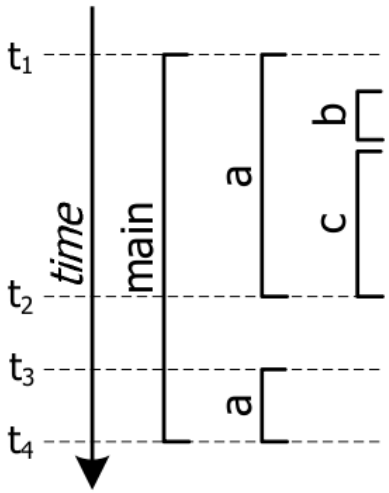
Profile tree

- every non-leaf node also has a leaf child called its residual node. This node represents the cost of running the body of code excluding the costs of the methods called by it.

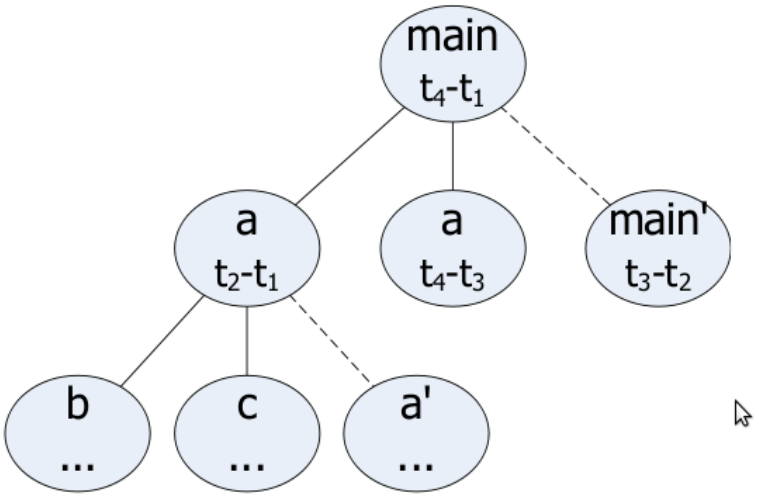
Profile tree

- every non-leaf node also has a leaf child called its residual node. This node represents the cost of running the body of code excluding the costs of the methods called by it.
- each edge is annotated with the state size at the time of invocation of the child node, plus the state size at the end of that invocation

Profile tree



(a) trace



(b) profile tree

Some math

- collect timings at method entry and exit points

Some math

- collect timings at method entry and exit points
- compute migration costs (edge weights) by simulating migration at each profiled method

Some math

- collect timings at method entry and exit points
- compute migration costs (edge weights) by simulating migration at each profiled method
  - perform the suspend-and-capture operation of the migrator

Some math

- collect timings at method entry and exit points
- compute migration costs (edge weights) by simulating migration at each profiled method
  - perform the suspend-and-capture operation of the migrator
  - measure the captured state size, both when invoking the child node and when returning from it

Some math

- collect timings at method entry and exit points
- compute migration costs (edge weights) by simulating migration at each profiled method
  - perform the suspend-and-capture operation of the migrator
  - measure the captured state size, both when invoking the child node and when returning from it
- fill the profile trees with appropriate values (some converting from size of heaps to time needed to migrate them etc.)

Energy consumption model

We can define also energy consumption model. Authors use a model that maps three system variables to a power level.

Energy consumption model

We can define also energy consumption model. Authors use a model that maps three system variables to a power level.

- CPU activity (processing/idle)

Energy consumption model

We can define also energy consumption model. Authors use a model that maps three system variables to a power level.

- CPU activity (processing/idle)
- display state (on/off)

Energy consumption model

We can define also energy consumption model. Authors use a model that maps three system variables to a power level.

- CPU activity (processing/idle)
- display state (on/off)
- network state (transmitting or receiving/idle)

Optimization Solver

Optimization Solver

is used to pick which application methods to migrate to the clone from the mobile device, so as to minimize the expected cost of the partitioned application

General mechanism

What we have?

General mechanism

What we have?

- particular execution E

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

What we want to do?

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

What we want to do?

Optimally replace annotations in T with those in T', so as to minimize the total node and weight cost of the hybrid profile tree.

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

What we want to do?

Optimally replace annotations in T with those in T', so as to minimize the total node and weight cost of the hybrid profile tree.

How to do it?

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

What we want to do?

Optimally replace annotations in T with those in T', so as to minimize the total node and weight cost of the hybrid profile tree.

How to do it?

- static analysis dictates the legal ways to fetch annotations from T' into T

General mechanism

What we have?

- particular execution E
- two profile trees T and T' for execution E

What we want to do?

Optimally replace annotations in T with those in T', so as to minimize the total node and weight cost of the hybrid profile tree.

How to do it?

- static analysis dictates the legal ways to fetch annotations from T' into T
- dynamic profiling dictates the actual trees T and T'

Translate objective world to math equations

- not all partitioning choices are legal - transform restriction to some relations

Translate objective world to math equations

- not all partitioning choices are legal - transform restriction to some relations
- define cost of a legal partition as some math expression (use costs calculated by dynamic profiler)

Translate objective world to math equations

- not all partitioning choices are legal - transform restriction to some relations
- define cost of a legal partition as some math expression (use costs calculated by dynamic profiler)
- finally the problem is to minimize the cost

Translate objective world to math equations

- not all partitioning choices are legal - transform restriction to some relations
- define cost of a legal partition as some math expression (use costs calculated by dynamic profiler)
- finally the problem is to minimize the cost

Note

Authors use a standard integer linear programming solver to solve this optimization problem with the above constraints.

Overview

The purpose of the distributed execution mechanism in CloneCloud is to implement a specific partition of an application process running inside an application-layer VM, as determined during partitioning.

General mechanism

The life-cycle of a partitioned application is as follows

General mechanism

The life-cycle of a partitioned application is as follows

- when the user attempts to launch a partitioned application, current execution conditions are looked up in a database of pre-computed partitions

General mechanism

The life-cycle of a partitioned application is as follows

- when the user attempts to launch a partitioned application, current execution conditions are looked up in a database of pre-computed partitions
- the application binary loads the partition and instruments the chosen methods with migration and re-integration points-special VM instructions

General mechanism

The life-cycle of a partitioned application is as follows

- when the user attempts to launch a partitioned application, current execution conditions are looked up in a database of pre-computed partitions
- the application binary loads the partition and instruments the chosen methods with migration and re-integration points-special VM instructions
- when execution of the process on the mobile device reaches a migration point, the executing thread is suspended and its state is packaged and shipped to a synchronized clone

General mechanism

- there, the thread state is instantiated into a new thread with the same stack and reachable heap objects, and then resumed

General mechanism

- there, the thread state is instantiated into a new thread with the same stack and reachable heap objects, and then resumed
- when the migrated thread reaches a re-integration point, it is similarly suspended and packaged as before, and then shipped back to the mobile device

General mechanism

- there, the thread state is instantiated into a new thread with the same stack and reachable heap objects, and then resumed
- when the migrated thread reaches a re-integration point, it is similarly suspended and packaged as before, and then shipped back to the mobile device
- finally, the returned packaged thread is merged into the state of the original process

Granulity

CloneCloud migration operates at the granularity of a thread.

- allows a multi-threaded process (e.g., a process with a UI thread and a worker thread)

Migrator thread

Migrator thread

The thread migrator is a native thread, operating within the same address space as the migrant thread, but outside the virtual machine. As such, the migrator has the ability to view and manipulate both native process state and virtualized state.

General mechanism

- Upon reaching a migration point, the job of the thread migrator is to suspend a migrant thread, collect all of its state, and pass that state to the node manager for data transfer.

General mechanism

- Upon reaching a migration point, the job of the thread migrator is to suspend a migrant thread, collect all of its state, and pass that state to the node manager for data transfer.
- Remaining threads continue processing, unless they attempt to access such marked (migrated state), in which case they block until the off-loaded thread returns.

General mechanism

- Upon reaching a migration point, the job of the thread migrator is to suspend a migrant thread, collect all of its state, and pass that state to the node manager for data transfer.
- Remaining threads continue processing, unless they attempt to access such marked (migrated state), in which case they block until the off-loaded thread returns.
- As soon as the captured thread state is transferred to the target clone device, the node manager passes it to the migrator of a freshly allocated process.

General mechanism

- Upon reaching a migration point, the job of the thread migrator is to suspend a migrant thread, collect all of its state, and pass that state to the node manager for data transfer.
- Remaining threads continue processing, unless they attempt to access such marked (migrated state), in which case they block until the off-loaded thread returns.
- As soon as the captured thread state is transferred to the target clone device, the node manager passes it to the migrator of a freshly allocated process.
- Reintegration is almost identical conceptually to the original migration (the context must update the original thread state to match the changes effected at the clone instead of creating new one)

State merge

What's the problem?

What to do with new objects? How to update existing objects? What with deleted objects? Object references themselves are not sufficient in that respect, since in most application-layer VMs, references are implemented as native memory addresses, which look different in different processes, across different devices and possibly architectures, and tend to be reused over time for different objects.

Object mapping table

The solution is object mapping table.

Object mapping table

The solution is object mapping table.

- use during state capture and reinstantation

Object mapping table

The solution is object mapping table.

- use during state capture and reinstantation
- stored while a thread is executing at a clone

Object mapping table

The solution is object mapping table.

- use during state capture and reinstantation
- stored while a thread is executing at a clone
- MID - unique ID give to object by mobile devie

Object mapping table

The solution is object mapping table.

- use during state capture and reinstantation
- stored while a thread is executing at a clone
- MID - unique ID give to object by mobile devie
- CID - unique ID give to object by clone

Object mapping table

How it works:

Object mapping table

How it works:

- while migration is initiated - mapping table is created and MID id filled for captured objects

Object mapping table

How it works:

- while migration is initiated - mapping table is created and MID id filled for captured objects
- clone assign CID for each object

Object mapping table

How it works:

- while migration is initiated - mapping table is created and MID id filled for captured objects
- clone assign CID for each object
- while reintegration

Object mapping table

How it works:

- while migration is initiated - mapping table is created and MID id filled for captured objects
- clone assign CID for each object
- while reintegration
  - objects with MID null was freshly created (must be allocated at mobile)

Object mapping table

How it works:

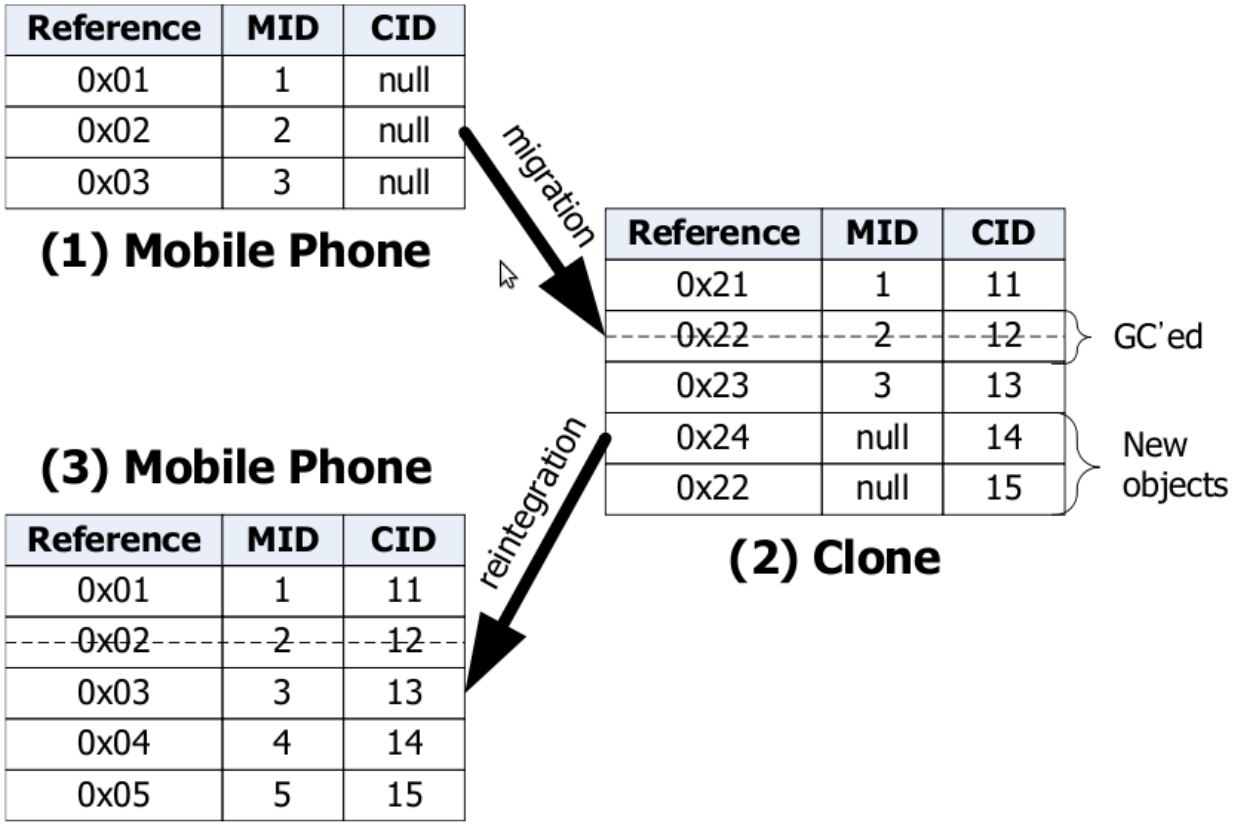
- while migration is initiated - mapping table is created and MID id filled for captured objects
- clone assign CID for each object
- while reintegration
  - objects with MID null was freshly created (must be allocated at mobile)
  - objects with MID and CID not nulls must be updated

Object mapping table

How it works:

- while migration is initiated - mapping table is created and MID id filled for captured objects
- clone assign CID for each object
- while reintegration
  - objects with MID null was freshly created (must be allocated at mobile)
  - objects with MID and CID not nulls must be updated
  - object deleted at the clone are ignored and no mapping is send back

Object mapping table



Technicals details

- Authors implemented prototype of CloneCloud partitioning and migration on the cupcake branch of the Android OS

Technicals details

- Authors implemented prototype of CloneCloud partitioning and migration on the cupcake branch of the Android OS
- Tested on HTC G1 device equipped with both WiFi and 3G connections

Technicals details

- Authors implemented prototype of CloneCloud partitioning and migration on the cupcake branch of the Android OS
- Tested on HTC G1 device equipped with both WiFi and 3G connections
- Clones running the Android x86 virtual machine

Technicals details

- Authors implemented prototype of CloneCloud partitioning and migration on the cupcake branch of the Android OS
- Tested on HTC G1 device equipped with both WiFi and 3G connections
- Clones running the Android x86 virtual machine
- Clones execute on a server with a 3.0GHz Xeon CPU, running VMware ESX 4.1

Evaluation

Tested with three imeplemented

Evaluation

Tested with three imeplemented

- virus scanner

Evaluation

Tested with three imeplemented

- virus scanner
- image search

Evaluation

Tested with three imeplemented

- virus scanner
- image search
- privacy-preserving targeted advertising

Evaluation

Static analysis

The static analysis took 23 seconds on average with these application.

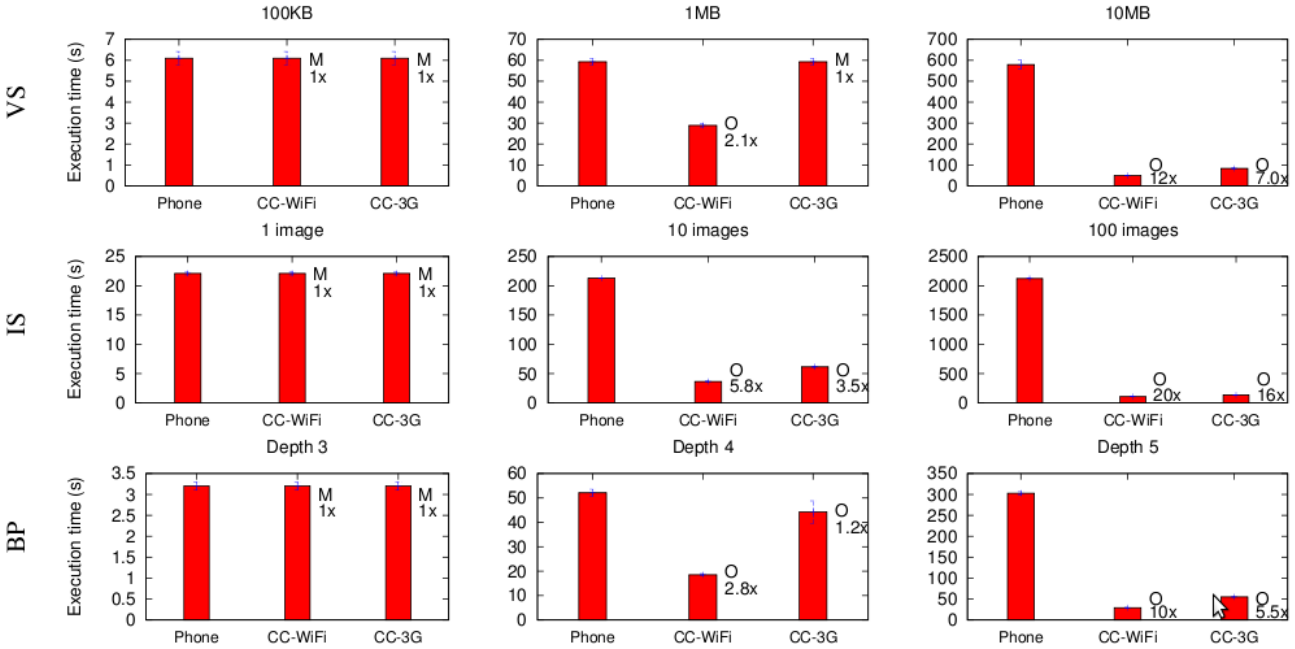
Dynamic profiler

Generating an optimizer (ILP) script from the profile trees, constraints and execution conditions, and solving the optimization problem, takes less than one second.

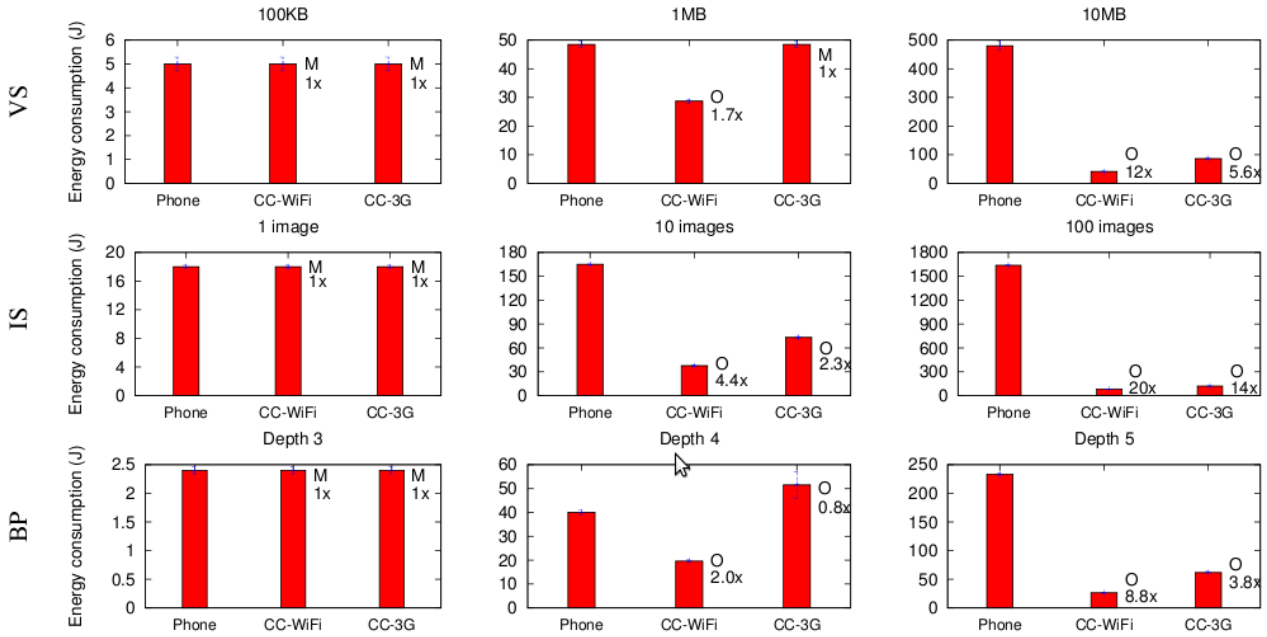
Execution time

Application	Input Size	Phone Exec. (sec) Mean (std)	Clone Exec. (sec) Mean (std)
VS	100KB	6.1 (0.32)	0.2 (0.01)
	1MB	59.3 (1.49)	2.2 (0.01)
	10MB	579.5 (20.76)	22.5 (0.08)
IS	1 img	22.1 (0.26)	0.9 (0.07)
	10 img	212.8 (0.44)	8.0 (0.03)
	100 img	2122.1 (1.27)	79.2 (0.44)
BP	depth 3	3.3 (0.10)	0.2 (0.01)
	depth 4	52.1 (1.45)	1.8 (0.07)
	depth 5	302.7 (3.76)	10.9 (0.19)

Time save


VS
IS
BP
100KB
1MB
10MB
1 image
10 images
100 images
Depth 3
Depth 4
Depth 5
Execution time (s)
Phone
CC-WiFi
CC-3G
M 1x
O 2.1x
O 12x
O 7.0x
O 5.8x
O 3.5x
O 20x
O 16x
O 2.8x
O 1.2x
O 10x
O 5.5x

Energy save


VS
IS
BP
100KB
1MB
10MB
1 image
10 images
100 images
Depth 3
Depth 4
Depth 5
Energy consumption (J)
Phone
CC-WiFi
CC-3G
M 1x
O 1.7x
O 12x
O 5.6x
O 4.4x
O 2.3x
O 20x
O 14x
O 0.8x
O 2.0x
O 8.8x
O 3.8x