CloneCloud: Elastic Execution between Mobile Device and Cloud, *Chun et al.*

Noah Apthorpe

Department of Computer Science
Princeton University

October 14th, 2015

Motivation

- Mobile applications are limited in computation speed, storage capacity, network communication rate, and available power
- Migrating portions of applications to the cloud can improve resource usage and overall performance
- However, most mobile/cloud applications are either
  - Monolithic processes with minimal cloud support
  - Client/server paradigms with nearly all computation on cloud
  - Tailored to specific mobile/cloud resources
- **Goal:** Automatic fine-grained application migration (partitioning) based on mobile/cloud conditions at runtime

Overview
System Model

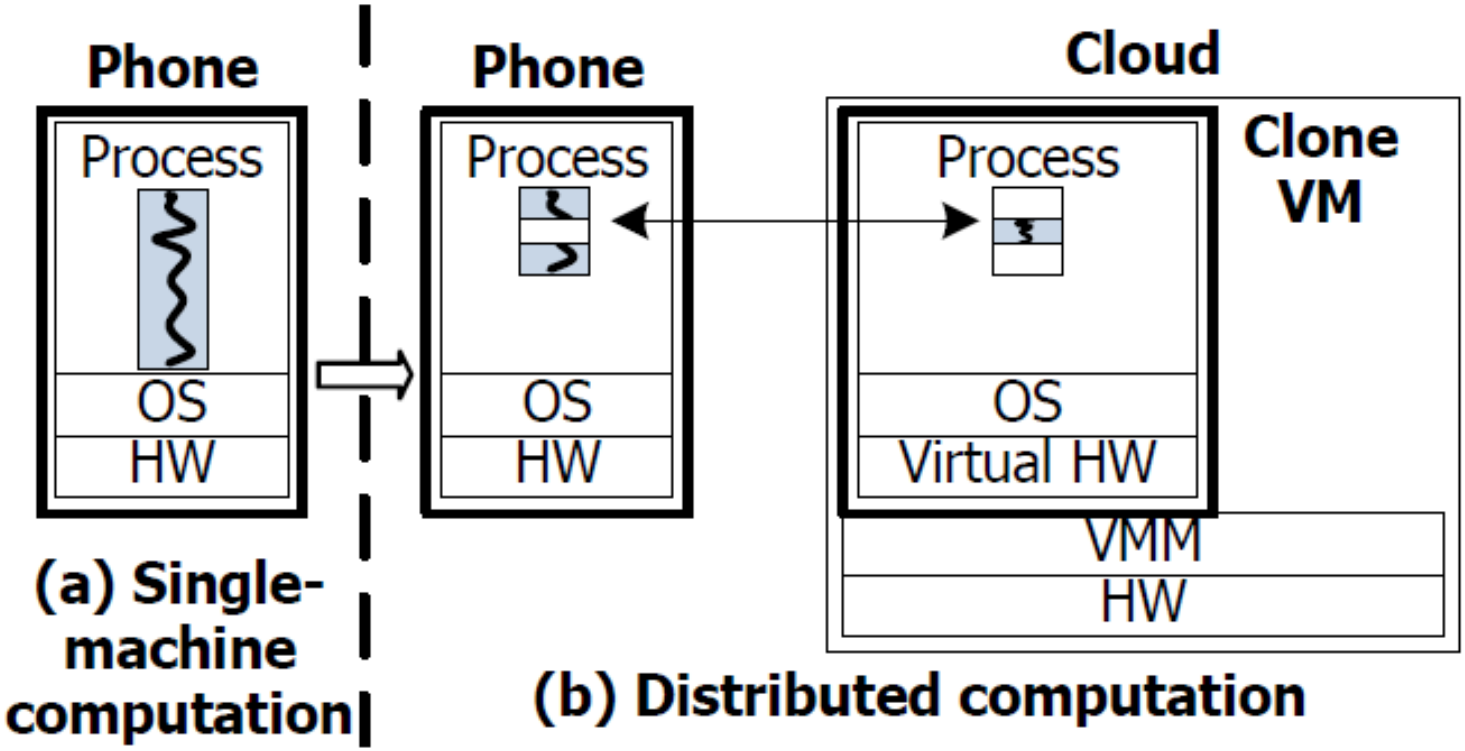


Figure 1. CloneCloud system model. CloneCloud transforms a single-machine execution (mobile device computation) into a distributed execution (mobile device and cloud computation) automatically.

Overview
Prototype Architecture

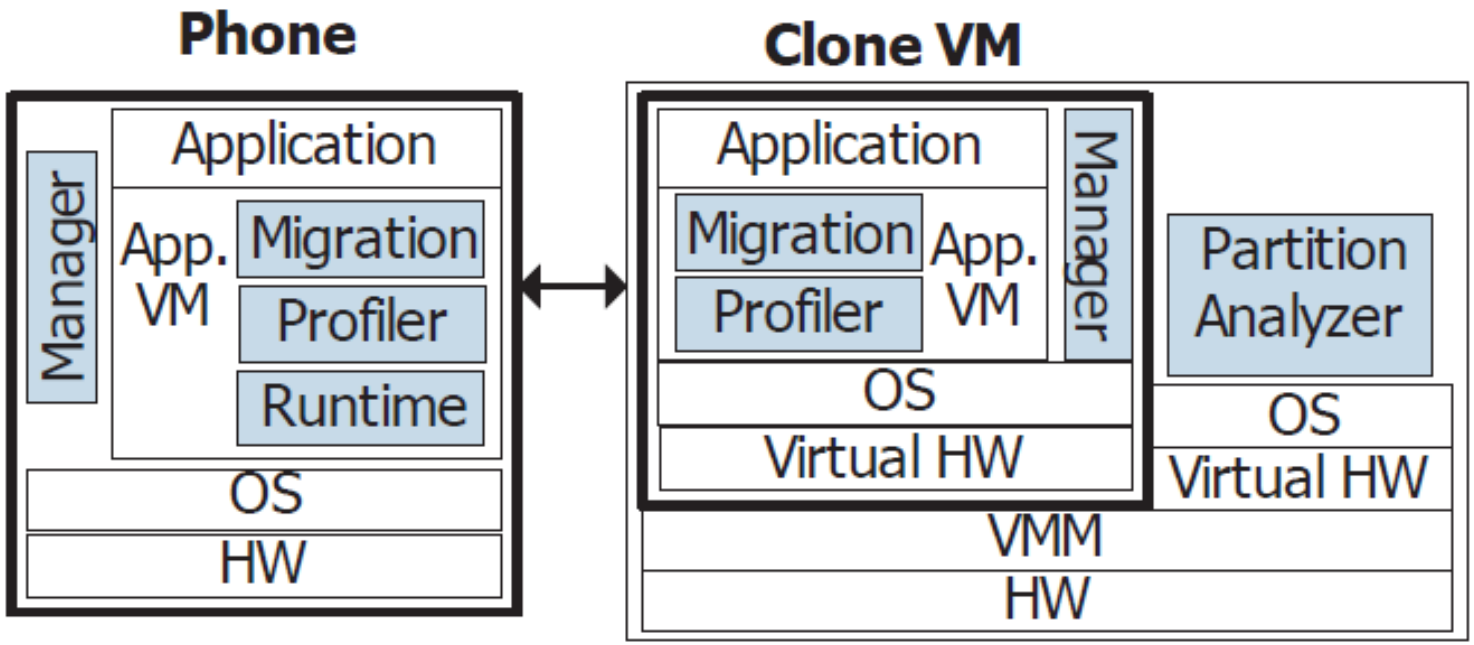


Figure 2. The CloneCloud prototype architecture.

Partitioning
Overview

- Which methods should be migrated to the cloud?
- Occurs off-line and without programmer interaction.
- Results in a database of possible partitions optimized for varying execution conditions
  - Network latency
  - Mobile CPU speed

  and different objective functions
  - Execution time
  - Power consumption
- A specific partition is selected at runtime

Partitioning
Overview

- 3 step partitioning implementation
  - Static analysis
    - Which possible partitions are legal?
  - Dynamic profiling
    - What behavior (execution time and power consumption) do application methods exhibit when run on the device and on the cloud
  - Optimization
    - Which legal partition optimizes the objective function(s) given behavior profiles

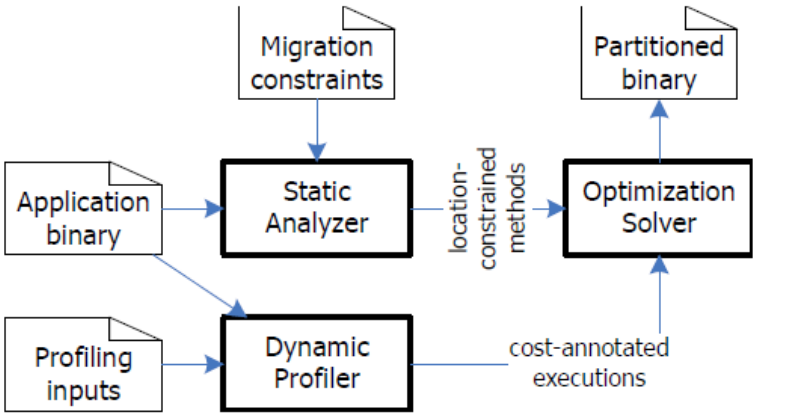


Figure 3. Partitioning analysis framework.

Partitioning
Static Analysis

- Control-flow graph created from code analysis
- Methods unable to be migrated are annotated
  - Methods accessing device-specific features (e.g. GPS or other sensors)
  - Methods sharing native (non-application VM) state
- Annotations are propagated upstream in graph
- Transitive calling relationships prevent nested migration

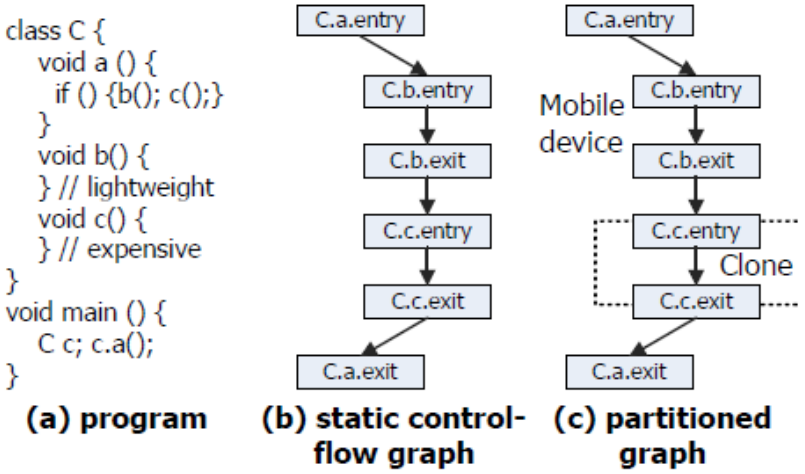


Figure 4. An example of a program, its corresponding static control-flow graph, and a partition.

Partitioning
Dynamic Profiling

- Application executed multiple times on device and on the cloud with randomly chosen arguments and UI events
- Profile trees of per-method costs (execution time and power usage) generated for each execution
- Power costs estimated from (CPU on/idle, Display on/off, network on/idle) tuples

- Computation cost $C_c(i, loc)$ and migration cost $C_s(i)$ calculated for each method invocation i.
  - $C_c(i, \text{cloud}) =$ residual cost of method i on the cloud
  - $C_s(i) =$ cost to suspend/resume thread + cost to transfer thread

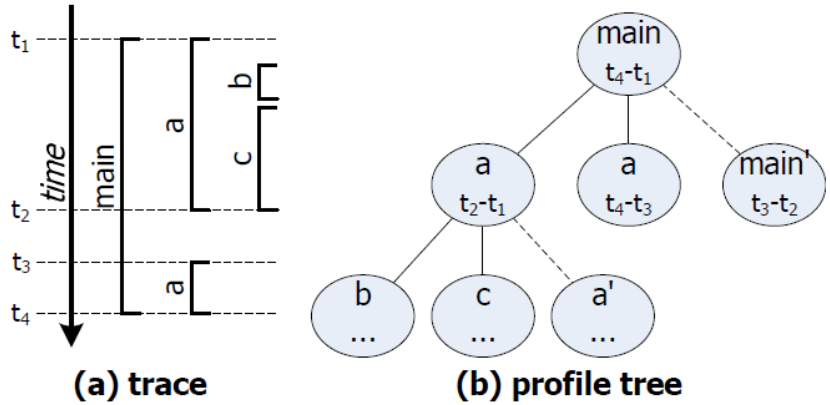
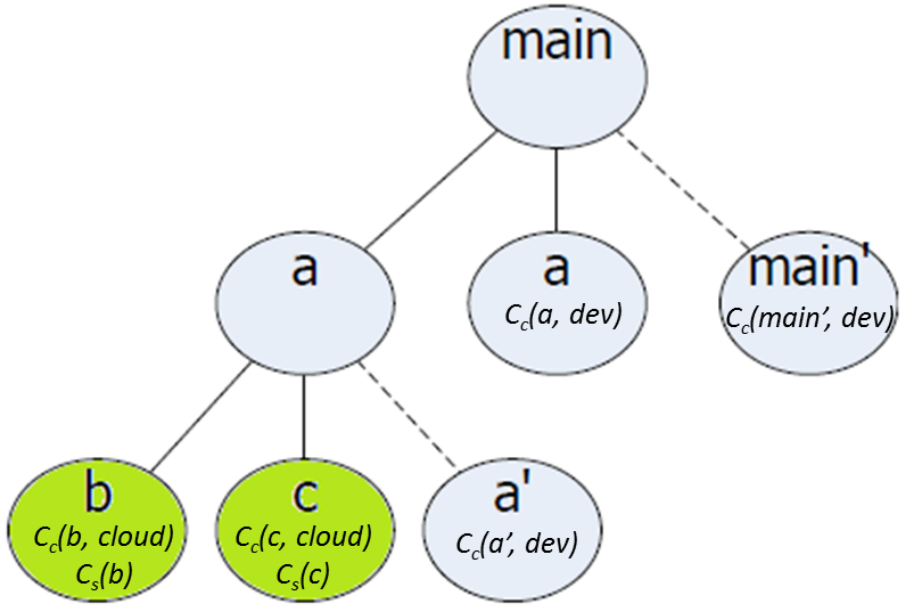


Figure 5. An example of an execution trace (a) and its corresponding profile tree (b). Edge costs are not shown.

Partitioning
Optimization

- "Optimally replacing annotations in T with those in T', so as to minimize the total node and weight cost of the hybrid profile tree"
  - $T =$ profile tree from executions on device
  - $T' =$ profile tree from executions on cloud

- Control flow graphs from static analysis merged with possible hybrid trees to find optimal legal partitions
- Optimization problem solved with integer linear programming solver



Distributed Execution
Migration Overview

- Partition selected from database based on runtime conditions
- Methods instrumented with migration and re-integration points
- Migrator thread suspends, packages, resumes, and merges thread state
- Node Manager coordinates clone provisioning/synchronization and device/cloud communication
- Migration at thread granularity allows true device/cloud parallelism

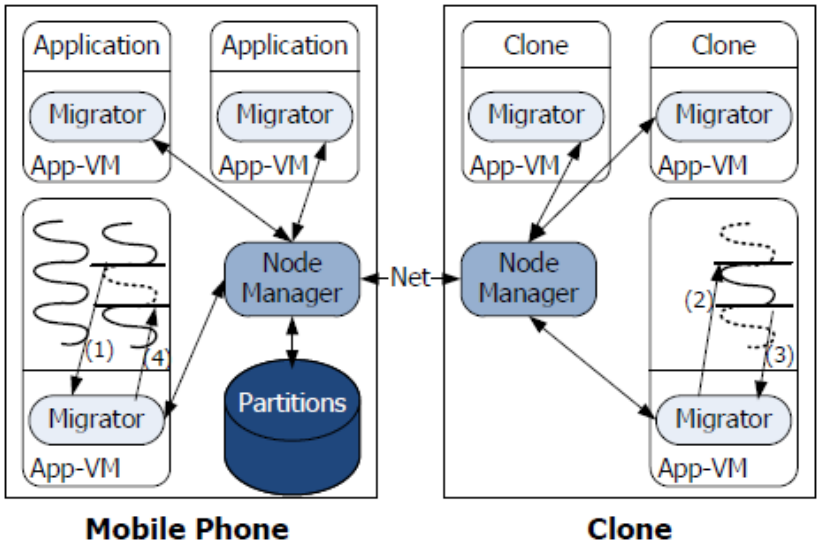


Figure 6. Migration overview.

Distributed Execution
State Transfer Details

- Migrator thread captures
  - execution stack frames
  - relevant data in process heap
  - register contents at migration point

  by crawling object references
- Captured state marked in memory and sent to cloud
- Object mapping table created to monitor object modification during remote execution
- Device state synchronized with state received from cloud at re-integration resume and merge

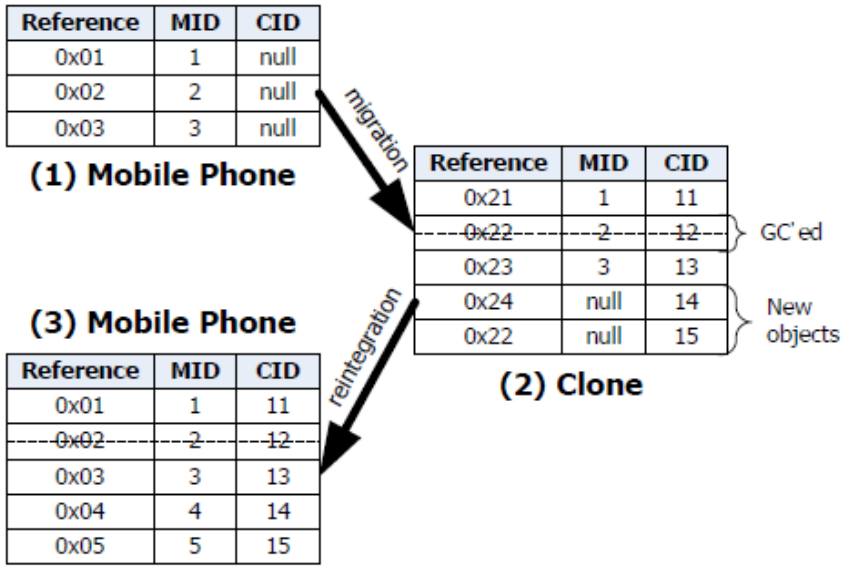


Figure 7. Object mapping example.

Evaluation
Speedup Opportunity

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)

Table 1. Execution times of virus scanning (VS), image search (IS), and behavior profiling (BP) applications, three input sizes for each. For each application and input size, the data shown include execution time at the phone alone and execution time at the clone alone.

Evaluation
Execution Time Analysis

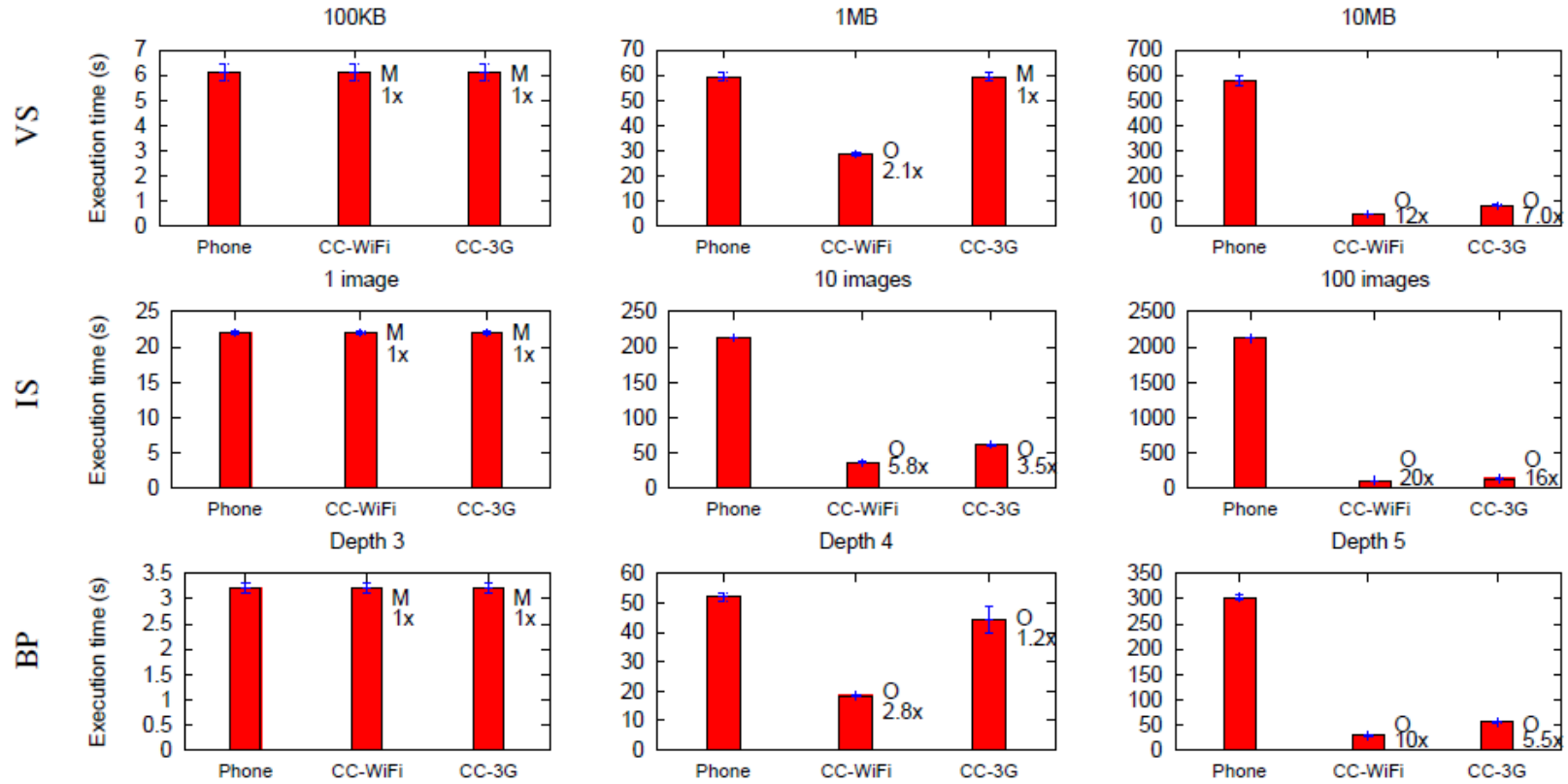


Figure 8. Mean execution times of virus scanning (VS), image search (IS), and behavior profiling (BP) applications with standard deviation error bars, three input sizes for each. For each application and input size, the data shown include execution time at the phone alone, that of CloneCloud with WiFi (CC-WiFi), and that of CloneCloud with 3G (CC-3G). The partition choice is annotated with M for "monolithic" and O for "off-loaded," also indicating the relative improvement from the phone-alone execution.

Evaluation
Power Consumption Analysis

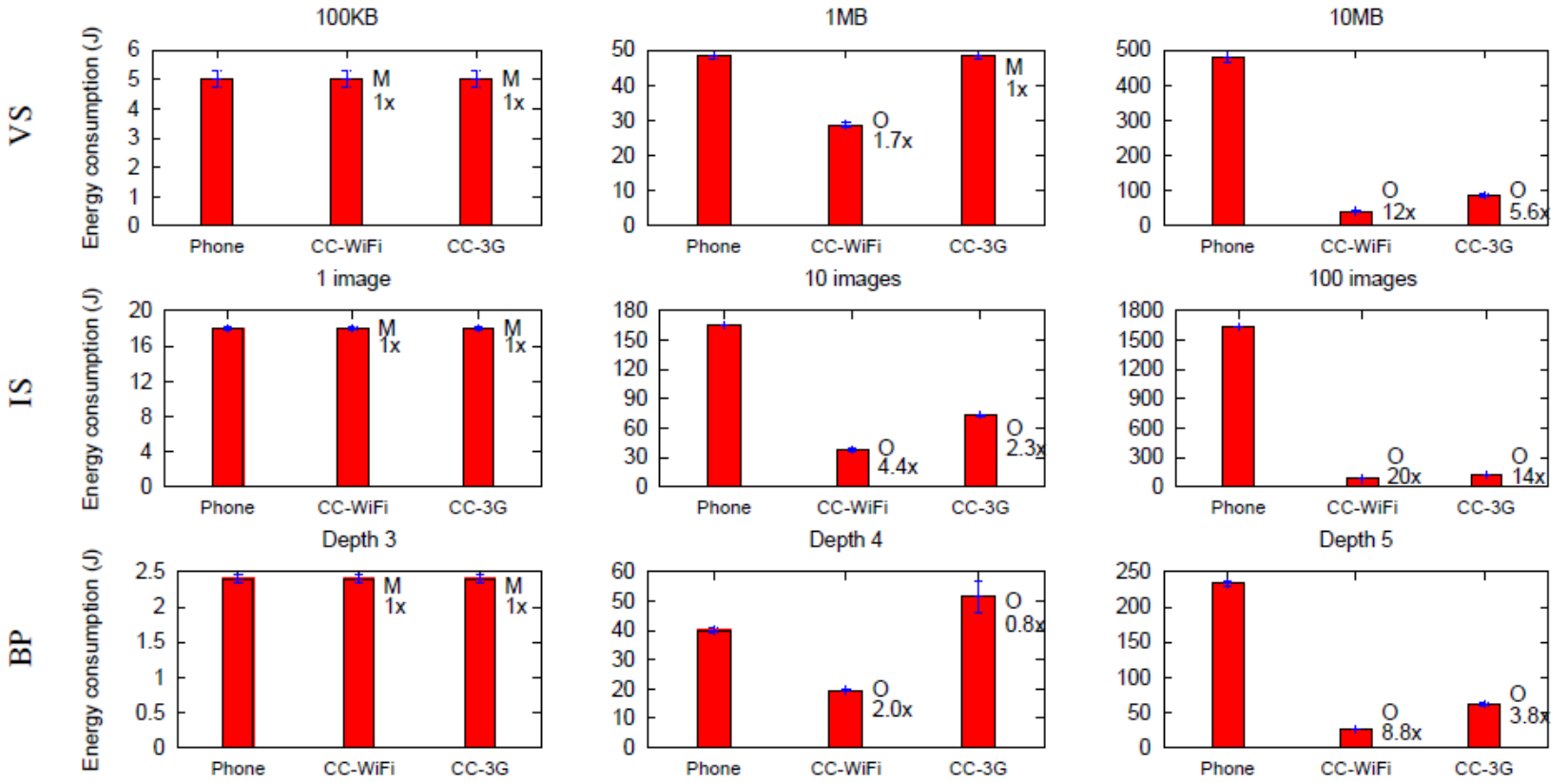


Figure 9. Mean phone energy consumption of virus scanning (VS), image search (IS), and behavior profiling (BP) applications with standard deviation error bars, three input sizes for each. For each application and input size, the data shown include execution time at the phone alone, that of CloneCloud with WiFi (CC-WiFi), and that of CloneCloud with 3G (CC-3G). The partition choice is annotated with M for "monolithic" and O for "off-loaded," also indicating relative improvement over phone-only execution.

Limitations
& Future Directions

- No distributed shared memory!
  - methods sharing memory must be co-located
  - threads accessing memory marked as "migrated" block until re-integration
- Cloud execution trusted implicitly
- Dynamic profiling only tests a sample of input space
- No migration at native method boundaries
- Can multiple threads be offloaded at once?
- Requires modifications to application VMs (e.g. Android Dalvik VM)
  - Platform-specific
  - iOS???
- No figure comparing CloneCloud apps to client/server versions

Discussion

Thoughts or questions?