Open System Laboratory of University of Illinois at Urbana Champaign presents:

IMCM: A Flexible Fine-Grained Adaptive Framework for Parallel Mobile Hybrid Cloud Applications

October 22, 2014

Reza Shiftehfar
PhD. Student and
Research Assistant in Computer Science
E-mail: sshifte2@illinois.edu

Kirill Mechitov
PhD. Student and
Post-Doctoral in Computer Science
E-mail: mechitov@illinois.edu

Gul Agha
Professor
in Computer Science
E-mail: agha@illinois.edu

Outline: A Fine-Grained Adaptive Framework for Mobile-Cloud Apps

- Outline:
  - Introduction
  - Motivation
  - Objective
  - Related Works
  - Approach
  - Conclusion

OPEN SYSTEMS LABORATORY

Introduction: An Adaptive Framework for Mobile-Cloud Apps

- Introduction
  - Mobile devices:
    - Ubiquitous and improving very fast
    - still constrained by limited resources:
      1. Weaker Hardware: (CPU, Memory, Storage)
      2. Restricted Network Access: (lower bandwidth, higher latency, cellular data usage charges/cap)
      3. Limited On-board Energy: (Battery-power restrictions)
  - Applications
    - are becoming more and more complex & resource-demanding
  - There is a gap between the two that will remain even in future

OPEN SYSTEMS LABORATORY

Introduction: An Adaptive Prog. Framework for Mobile-Cloud Apps

- Introduction
  - Cloud Computing:
    - allows efficient storage and processing of very large datasets
      - Efficient both in terms of time(speed) and cost(price)
    - provides elastic on-demand access to unlimited resources at affordable price
    - Serves as the center of the data storage and processing of many organizations
    - dominant term for new computing paradigm
  - Cloud has the potential to help with mobile restrictions:
    1. Remove constraints on mobile device hardware
    2. Reduce mobile energy consumption
  - How to do this? Use code-offloading

OPEN SYSTEMS LABORATORY

Introduction: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Introduction
  - *Code-Offloading:*
    - *Sending computation to more resource-full machines and bringing back the results*
  - *How to make code offloading happen:*
    1. *Ask programmers to rewrite the code*
       1. *Pros: Fine-grained*
       2. *Cons: Too much additional work, continuous maintenance*
    2. *Use full process or full VM migration*
       1. *Pros: No additional tasks for programmers*
       2. *Cons: Very coarse-grained, expensive*
    3. *Combine the two*

References: [1]

OPEN SYSTEMS LABORATORY

Introduction: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Introduction
  - *Current mobile application architectures:*
    1. *Offline model: (Native App.) (Fat-Client)*
       - *All parts installed and executed locally on the phone*
       - *Even required remote data is downloaded periodically*
    2. *Client-server model: (Web App.) (Thin-client)*
       - *Mobile provides minimum interface to connect to remote*
       - *Example: Using mobile browser to access Facebook*
  - *A third category is needed in between:*
    - *To create a flexible fine-grained adaptive solution*
    - *Allow dynamic component move between mobile device and cloud space*
    - *This is known as Mobile-Cloud Computation (MCC)*

OPEN SYSTEMS LABORATORY

Motivation: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Motivation
  - *Mobile-Cloud Computing (MCC) is challenging because of:*
    1. *Applications:*
       - *Have different requirements*
       - *Behave differently at different times*
       - *Have different applications goals:*
         1. *Maximizing the performance*
         2. *Minimizing network data usage*
         3. *Minimizing mobile energy consumption*
    2. *Mobile devices:*
       - *Different hardware capabilities*
       - *Different quality of network access and limitations*
       - *Different amount of accessible energy to use*

OPEN SYSTEMS LABORATORY

Motivation: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Motivation
  - *Mobile-Cloud Computing (MCC) is challenging because of: (cont.)*
    3. *Security and Privacy:*
       - *Current Cloud solutions:*
         1. *Typically provide very limited security protection*
         2. *Traditionally Assumed as a trusted environment*
       - *This can be true when:*
         1. *Cloud is kept separate from the outside word*
         2. *only used by authorized people*
       - *Public cloud:*
         - *owned and operated by 3rd party companies*
         - *Risky for moving sensitive or confidential data or computation*

OPEN SYSTEMS LABORATORY

Motivation: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Motivation
  - *Mobile-Cloud Computing (MCC) is challenging because of: (cont.)*
    4. *Users:*
       - *have different expectations at different time*
       - *might have further concerns about the privacy of their data.*
  - *Mobile Applications are used for different target goals in different environmental conditions with dynamic requirements and expectations*

OPEN SYSTEMS LABORATORY

Motivation: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Motivation:
  - *Mobile-Cloud application requirements:*
    - *Need to be consisted of proper independent components*
    - *Components need to be dynamically managed and moved around*
    - *Component distribution must be:*
      1. *Automatic*
      2. *Flexible*
      3. *Fine-grained*
      4. *Adaptive*

OPEN SYSTEMS LABORATORY

Objective: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

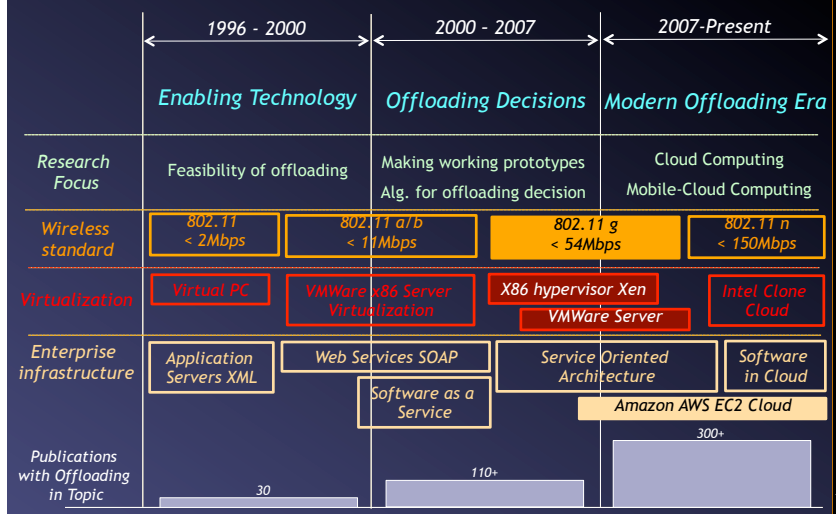
- Bridge the gap between mobile application development, cloud computing and dynamic adaptive code-offloading
- Separate application logic from application component configuration and distribution plan
- Allow dynamic distribution of application components in response to run-time environmental changes while preserving required constraints
- Provide Policy-driven offloading adjustment for Cost, Privacy, Quality

Mobile-cloud applications require restricted fine-grained adaptive and dynamic configuration and distribution of code and data components. Such adaptation can be achieved by online monitoring of user context, resources, communications and solving optimization to update component configuration and distribution

OPEN SYSTEMS LABORATORY

History: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Offloading History:



References: [1, 2, 3, 9, 10]

OPEN SYSTEMS LABORATORY

Cloud Model

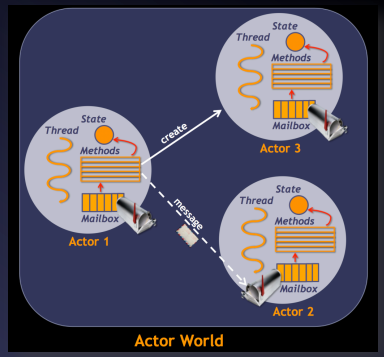
- Traditionally, Cloud infrastructure is provided by third-party large organizations and is known as Public Cloud.
  - Problems when stored by 3rd part:
    1. Potential lack of control and transparency
    2. Legal implications of data and applications
- This encouraged companies to build their own private Clouds
  - Problems:
    - Not as efficient/scalable/elastic as public Cloud
- Recently, Hybrid Cloud:
  - Benefits from all advantages of public Cloud
  - Keeps confidential or sensitive data or algorithms in-house

Cloud Application Model

- Mobile-Cloud Application:
  - Should have independent components that can be distributed over public Cloud, private Cloud, and the end-user device.
  - These components can
    - store data or
    - perform computation
  - To maximize parallelism:
    - avoid global or shared state
    - Limit interactions to communicating using messages
- This aligns well with the Actor Model of Computation

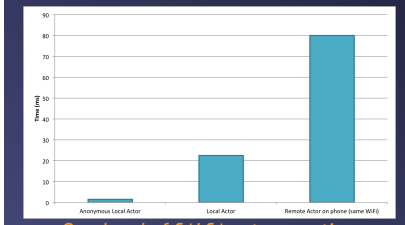
Cloud Application Model: Actor Model of Computation

- Advantages:
  1. Natural Concurrency
  2. No shared state (No data race)
  3. Elasticity
  4. Decentralization
  5. Ease of scaling-up or -out
  6. Location Transparency
  7. Transparent Migration

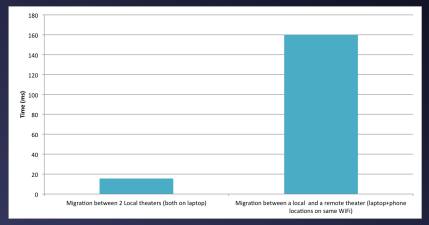


Cloud Application Model: Actor Model of Computation

- We used SALSA:
  - Loyal to standard actor semantics
  - Depends on Java -> can be made to work on Android DalvikVM
  - Provides lightweight actors
  - Lightweight actors -> migration between devices is fast



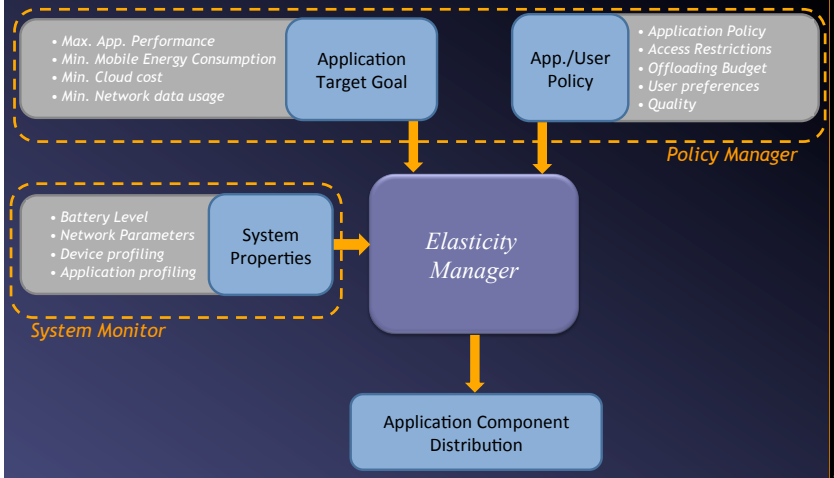
Overhead of SALSA actor creation



Overhead of SALSA actor migration

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Overall System Design:



OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- First Step toward Application Offloading:
  - *Ignoring Offloading process overhead:*
    - *Running the same code on a faster machine provides linear speedup:*

$$Speedup = \frac{S_s}{S_m}$$

    - S_s , S_m : *the speed of server and mobile device*

$$Speedup = \frac{S_s}{S_m} = \frac{F_{server} * C_{server} * X_{server}}{F_{mobile} * C_{mobile}}$$

    - F_{server} , F_{mobile} : *Freq. of the server and mobile processor*
    - C_{server} , C_{mobile} : *No. of cores available on the server and mobile*
    - X_{server} : *Additional speedup resulting from availability of more caches and more memory in addition to more aggressive pipelining*

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- System Resources and Application Performance:
  - *Experimental results:*

	Remote Server	Mobile Device
System	Macbook Pro - Retina Mid 2012	Samsung Google Nexus S
OS	Mac OSX 10.9.4	Android 4.1.2
VM	JVM (JRE 1.6)	DalvikVM
Processor	Intel Core i7	ARM Coretex-A8
Processor speed	2.3 GHz	1 GHz
Number of cores	4	1
L2 Cache(per core)	256 KB	256 KB
L3 Cache	6MB	-
Memory	16 GB	512 MB

$$Speedup = \frac{S_s}{S_m} = \frac{F_{server} * C_{server} * X_{server}}{F_{mobile} * C_{mobile}} = \frac{2.3 * 4 * 7}{1.0 * 1} = 64$$

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- System Resources and Application Performance:
  - *Detailed experimental results:*

Experiment	Description	Application Characteristic			
		Comp.	Comm.	I/O	
				read	write
NQueen	Places N Queens on N*N board	intensive	-	-	-
Image	Detects & recognizes all faces in a photo	intensive	limited	limited	-
Trap	Uses trapezoidal rule to calculate definite integral	intensive	limited	-	-
Virus	Scans a file stream for a specific virus signature	-	-	intensive	-
Rotate	Reads, rotates & saves an image to disk	-	-	intensive	intensive
ExSort	External Sort of the content of a file	intensive	-	intensive	intensive
Heat1	simulates heat exchange on a board	limited	medium	-	-
Heat2	simulates heat exchange on a board	limited	high	-	-

Experiment	Raw Speedup	Offload Speedup
NQueen	73	56
Image	91	44
Trap	30	21
Virus	28	21
Rotate	28	9
ExSort	46	36
Heat1	31	29
Heat2	14	14

OPEN SYSTEMS LABORATORY

Second Step toward Application Offloading:

- Including Offloading process overhead:
- Code offloading for Maximizing the Application Performance:

$$\frac{w}{S_m} > \frac{d_i}{B} + \frac{w}{S_s}$$

  - w: amount of computation to offload
  - S_m, S_s: Speed of mobile and remote server processor
  - B: the network bandwidth
  - d_i: size of data to be transferred
  - Offloading is beneficial, when left hand side > right hand side

Second Step toward Application Offloading: (ctd.)

$$w * (\frac{1}{S_m} - \frac{1}{S_s}) > \frac{d_i}{B}$$

- It is true for:
  - Large w -> program requires heavy computation
  - Large S_s -> server is fast enough
  - Small d_i -> small amount of data to be transferred
  - Large B -> available bandwidth is large

$$\frac{w}{S_m} > \frac{d_i}{B}$$

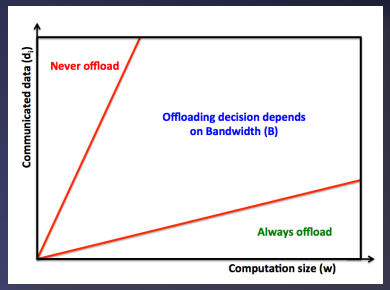
- If this is not true, offloading is not economical even with $S_s = \infty$
- So, only tasks with following conditions worth offloading:
  - Heavy computation (large w)
  - Small data exchange (small di)

Second Step toward Application Offloading: (ctd.)

- Our experimental setup:

$$w * (\frac{1}{1024Mhz} - \frac{1}{2.3 * 1024MHz * 4 * 7}) > \frac{d_i}{B} \rightarrow B > 1040 * \frac{d_i}{w}$$

  - B_{min} is the min. required bandwidth for offloading to be economical
  - Also depends on d_i/w



Code offloading for Maximizing Application Performance:

- N-Queen problem:
  - Input: Integer value N -> d_i is very small
  - Problem is computationally expensive -> w is very large
  - B_{min} = 1040 * 1/16 = 65 bit/sec

Technology	Average Existing Bandwidth (bit/sec)
2G (Edge)	0.2 Mega
3G	1 Mega
4G (LTE)	5 Mega
WiFi	10 Mega

  - So, it is always efficient to offload N-Queen problem

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading for Maximizing Application Performance:
  - *Image Processing Problem*:
    - *Input*: *Reasonable size of picture (w medium)*
    - *Assuming that remote server is super fast (Ss = ∞)*
    - *Offloading depends on d_i and B*
    - *Case 1*: *Face detection part is offloaded*
      - *Whole image has to be sent -> d_i is large*
      - *Then, Offload only if B is large*
    - *Case 2*: *Face detection is local*
      - *only extracted features needs to be sent over -> d_i is small*
      - *Then, Offload even for smaller B*

OPEN SYSTEMS LABORATORY

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Parameters affecting Offloading Decision:

1. *Problem Size:*
   - *N-Queen Problem:*

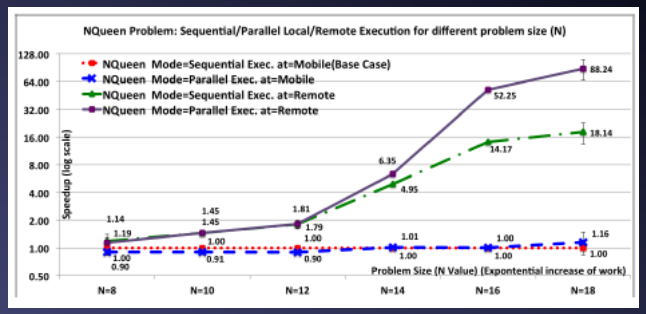


Figure: Speedup summary for different amount of work

OPEN SYSTEMS LABORATORY

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Parameters affecting Offloading Decision:

1. *Problem Size*
   - *Image Processing Problem:*

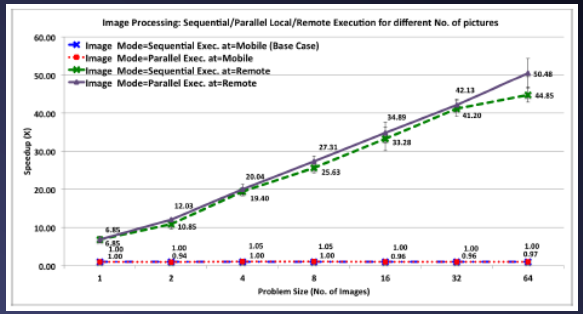


Figure: Speedup summary for different amount of work

OPEN SYSTEMS LABORATORY

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Parameters affecting Offloading Decision:

2. *Parallelism Degree of the problem*
   - *N-Queen Problem:*

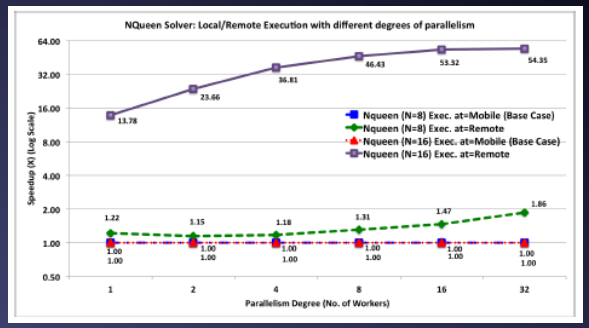


Figure: Speedup summary with different degree of parallelism

OPEN SYSTEMS LABORATORY

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading for Maximizing Application Performance:

2. *Parallelism Degree of the problem*
   - *Image Processing*
   - *Parallelism is great but ONLY if enough resources are available*

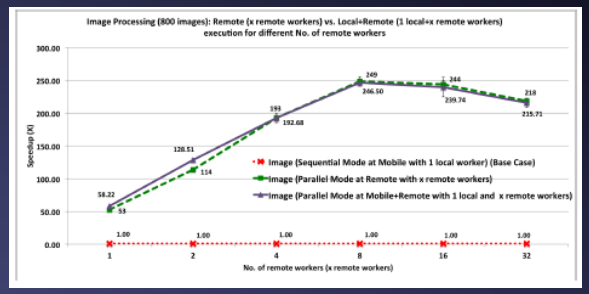


Image Processing: Local + Remote execution with different no. of remote workers

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading: Min. Mobile Energy Consumption
  - *Max performance case:*

$$\frac{w}{S_m} > \frac{d_i}{B} + \frac{w}{S_s}$$

  - *Min. Mobile Energy Consumption:*

$$P_{calc-mobile} * \frac{w}{S_m} > P_{comm-mobile} * \frac{d_i}{B} + P_{idle-mobile} * \frac{w}{S_s}$$

  - $P_{calc\text{-}mobile}$: *Mobile power consumption when performing comp.*
  - $P_{comm\text{-}mobile}$: *Mobile power consumption when communicating data over network*
  - $P_{idle\text{-}mobile}$: *Mobile power consumption when idle*
  - *Left hand side: energy consumed to perform comp. locally*
  - *Right hand side: energy consumed on the mobile to offload and then wait for the result to come back*
  - *Focus: Minimizing MOBILE energy consumed and not total energy*

References: [1]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading: Min. Mobile Energy Consumption
  - *Max performance case:*

$$w * \left(\frac{1}{S_m} - \frac{1}{S_s}\right) > \frac{d_i}{B}$$

  - *Min. Energy Consumption:*

$$w * \left(\frac{P_{calc-mobile}}{S_m} - \frac{P_{idle-mobile}}{S_s}\right) > P_{comm-mobile} * \frac{d_i}{B}$$

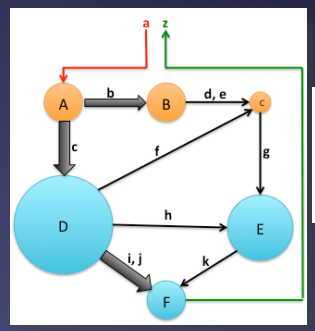
  - *Very similar:*
    - *Both assume phone to be in idle state while offloaded code is being executed*
    - *This assumption is only true for sequential applications*
    - *This is why previous research had similar offloading decision plan for both Min. Energy and Max. Performance*

References: [1, 3]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading in Large Applications:
  - *Previous analysis is for deciding on offloading one piece of code with w amount of computation*
  - *It is beneficial to offload multiple parts (methods) at a time.*
  - *This requires considering communication between components*



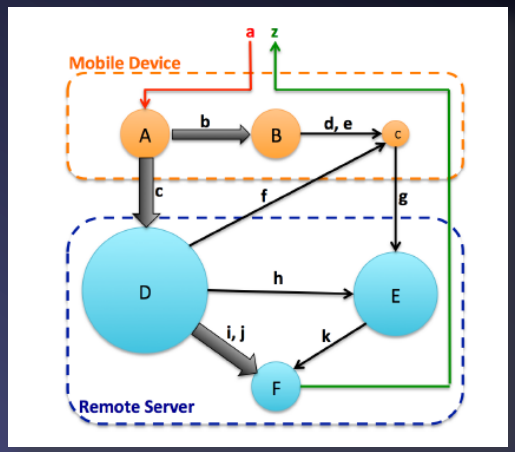
Relationship	Size of Input	Size of output
(b,c) = A(a)	a is small	b and c are large
(d,e) = B(b)	b is large	d and e are small
g = C(d,e)	d and e are small	b is small
(f,h,i,j) = D(c)	c is large	f & h are small, i & j are large
k = E(g,h)	g & h are small	k is small
z = F(i,j,k)	k is small, i & j are large	z is small

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading in Large Applications:
  - *A possible result for partitioning the program graph:*



References: [9,10]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- How Parallelism can affect Large Application Offloading?
  - *Local code execution is paused in all previous offloading scenarios*
    - *MAUI explicitly pauses the mobile code*
    - *ThinkAir and CloneCloud supports opportunistic parallelism*
      - *Results in Serial execution in practice*
      - *Their results for Max. Performance is same as Min. Mobile Energy Consumption*
  - *Keeping mobile phone in IDLE state while offloading wastes local resources.*
    - *Having local phone execute code in parallel makes Max. Performance results very different from Min. Mobile Energy Consumption*

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- How Parallelism can affect Large Application Offloading?
  - *Full-Parallelism mean:*
    - *Multiple Concurrent Components*
    - *Multiple Cloud resources*
    - *Simultaneous Local + Remote Execution*

Image Processing: Speedup from Local + Remote execution with different problem size

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading and Parallelism:
  1. *Maximizing Application Performance:*

$$
\begin{aligned}
Max(Application\ performance) &= Min(Total\ application\ execution\ time) \\
&= Min(\ Max(execution\ time\ at\ different\ locations) + \\
&\qquad Max(communication\ time\ between\ different\ locations)) \\
&= Min(\ Max_{0 \le L \le M}(execution\ time\ at\ Location\ L) + \\
&\qquad Max_{0 \le L \le M}(communication\ time\ from\ components\ on\ L\ to\ others))
\end{aligned}
$$

$$
\begin{aligned}
&subject\ to: \\
&\quad Loc(i,t_1) \epsilon LocAllowed(i,t_1)\ :\ \forall i \epsilon [1,N] \\
&\quad Loc(i,t_2) \epsilon LocAllowed(i,t_2)\ :\ \forall i \epsilon [1,N] \\
&\quad \sum_{i=1}^{N} LocEQ(L, Loc(i,t_2)) * (1 - LocEQ(L, Loc(j,t_1))) \le \alpha * Cores(L)\ :\ \forall L \epsilon [0,M]
\end{aligned}
$$

$$
LocAllowed(i,t) = \{x \epsilon [0,M] \mid component\ i\ is\ allowed\ to\ be\ placed\ at\ x\ at\ time\ t\}
$$

References: [ ]

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- **Code offloading and Parallelism:**
  - *Maximizing Application Performance:*

$$
\begin{aligned}
Max(Application\ performance) &= Min(Total\ application\ execution\ time) \\
&= Min(\ Max(execution\ time\ at\ different\ locations) + \\
&\qquad Max(communication\ time\ between\ different\ locations)) \\
&= Min(\ Max_{0 \le L \le M}(execution\ time\ at\ Location\ L) + \\
&\qquad Max_{0 \le L \le M}(communication\ time\ from\ components\ on\ L\ to\ others))
\end{aligned}
$$

$$
Max_{0 \le L \le M}(execution\ time\ at\ Location\ L) \ = \ \max_{0 \le L \le M} \frac{1}{Cores(L)} * \sum_{i=1}^{N} LocEQ(L, Loc(i,t_2)) * Exec(i, Loc(i,t_2)) * JobNo(i)
$$

$$
Max_{0 \le L \le M}(communication\ time\ from\ components\ on\ L\ to\ others) \ = \ \max_{0 \le L \le M} \frac{1}{B(L)} * \sum_{i=1}^{N} \sum_{j=1}^{N} LocEQ(L, Loc(i,t_2)) * (1 - LocEQ(L, Loc(j,t_2))) * profiledComm(i,j)
$$

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- **Code offloading and Parallelism:**
  2. *Minimizing Mobile Device Energy Consumption:*

$$
\begin{aligned}
&Min(Application\ Mobile\ Energy\ Consumption) \\
&\quad = Max(\ Energy\ Saving\ on\ Mobile\ Device) \\
&\quad = Max(\ Total\ Mobile\ Energy\ Saving\ by\ running\ components\ remotely \\
&\qquad - Energy\ Loss\ due\ to\ prev.\ local\ comm.\ becomming\ remote\ comm. \\
&\qquad + Energy\ Saving\ due\ to\ prev.\ remote\ comm.\ becomming\ local\ comm.)
\end{aligned}
$$

$$
\begin{aligned}
&subject\ to: \\
&\quad Loc(i,t_1) \epsilon LocAllowed(i,t_1) \ : \ \forall i \epsilon [1,N] \\
&\quad Loc(i,t_2) \epsilon LocAllowed(i,t_2) \ : \ \forall i \epsilon [1,N] \\
&\quad (Total\ Execution\ time\ after\ offloading) \le \\
&\qquad \beta * (Total\ execution\ time\ for\ running\ all\ components\ locally\ on\ the\ phone)
\end{aligned}
$$

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- **Code offloading and Parallelism:**
  - *Minimizing Mobile Device Energy Consumption:*

$$
\begin{aligned}
&= Max(\ Total\ Mobile\ Energy\ Saving\ by\ running\ components\ remotely \\
&\qquad - Energy\ Loss\ due\ to\ prev.\ local\ comm.\ becomming\ remote\ comm. \\
&\qquad + Energy\ Saving\ due\ to\ prev.\ remote\ comm.\ becomming\ local\ comm.)
\end{aligned}
$$

$$
Total\ Mobile\ Energy\ Saving\ by\ running\ components\ remotely = \sum_{i=1}^{N} LocEQ(0, Loc(i,t_1)) * (1 - LocEQ(0, Loc(i,t_2))) * Energy(i)
$$

$$
Energy\ Loss\ due\ to\ prev.\ local\ comm.\ becomming\ remote\ comm. = \sum_{i=1}^{N} LocEQ(0, Loc(i,t_2)) * (1 - LocEQ(0, Loc(j,t_2))) * ProfiledComm(i,j) * P_{commMobile}
$$

$$
Energy\ Saving\ due\ to\ prev.\ remote\ comm.\ becomming\ local\ comm. = \sum_{i=1}^{N} LocEQ(0, Loc(i,t_1)) * (1 - LocEQ(0, Loc(j,t_1))) * ProfiledComm(i,j) * P_{commMobile}
$$

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- **Code offloading and Parallelism:**
  - *Minimizing Mobile Device Energy Consumption:*

$$
\begin{aligned}
&subject\ to: \\
&(Total\ Execution\ time\ after\ offloading) \le \\
&\quad \beta * (Total\ execution\ time\ for\ running\ all\ components\ locally\ on\ the\ phone)
\end{aligned}
$$

$$
\begin{aligned}
&\max_{0 \le L \le M} \frac{1}{Cores(L)} * \sum_{i=1}^{N} LocEQ(L, Loc(i,t_2)) * Exec(i, Loc(i,t_2)) * JobNo(i) \\
&+ \max_{0 \le L \le M} \frac{1}{B(L)} * \sum_{i=1}^{N} \sum_{j=1}^{N} LocEQ(L, Loc(i,t_2)) * (1 - LocEQ(L, Loc(j,t_2))) * profiledComm(i,j) \\
&\le \beta * \frac{1}{Cores(L)} \sum_{i=1}^{N} Exec(i,0)
\end{aligned}
$$

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading and Parallelism:Max. App. Perf. In Parallel
  - *Evaluation of Elasticity Manager:*

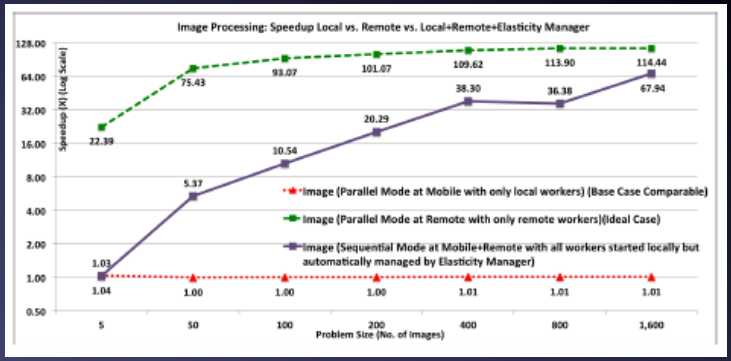


Image proc: Local exec (base case) vs. Remote Exec (Ideal Case) vs. Automatic Management (All components starting locally and then distributed)

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Code offloading & Parallelism: Max. App. Perf. In Parallel
  - *Overhead of Elasticity Manager:*

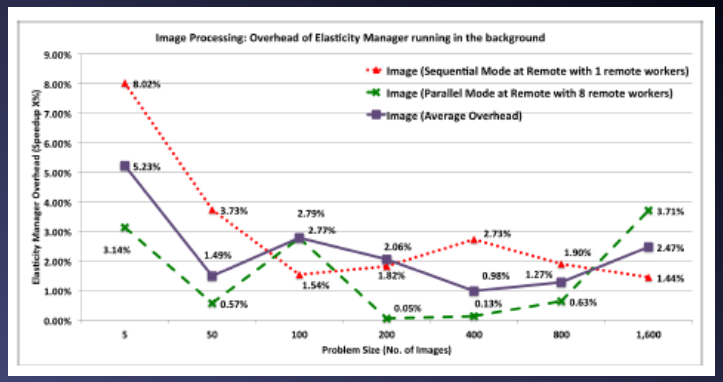


Image proc.: Overhead resulting from Elasticity Manager

References: [ ]

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Policy-based offloading adjustment:
  - *Offloading with unlimited offloading budget:*

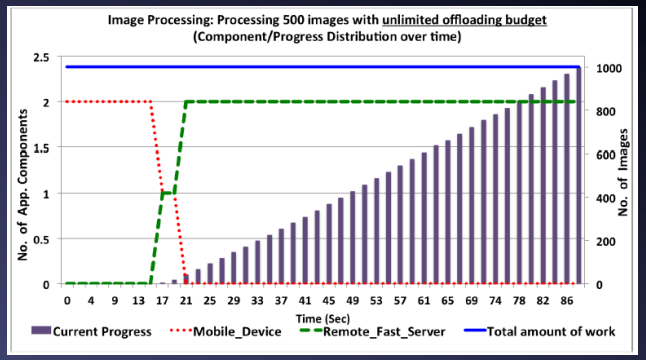


Image proc: Automatic mobile code offloading to a single remote server without any offloading budget restriction

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Policy-based offloading adjustment:
  - *Offloading with a fixed total offloading budget:*

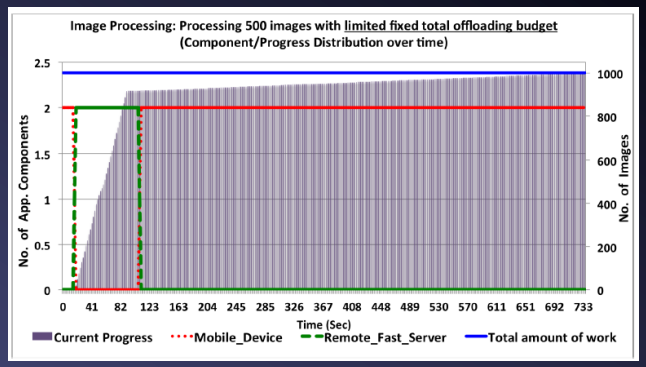


Image proc: Automatic mobile code offloading to a single remote server with a fixed total offloading budget restriction

OPEN SYSTEMS LABORATORY

Approach: *An Adaptive Prog. Framework for Mobile-Cloud Cloud Apps*

- Policy-based offloading adjustment:
  - *Offloading with a limited offloading budget rate:*

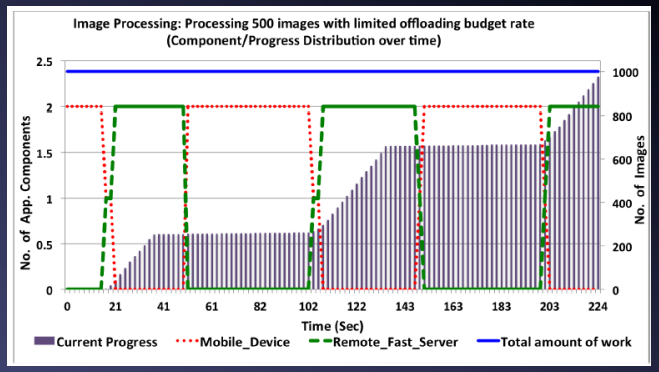


Image proc: Automatic mobile code offloading to a single remote server with a limited offloading budget rate per time interval

OPEN SYSTEMS LABORATORY

Conclusion: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

- Conclusion:
  - A Flexible Fine-Grained Adaptive Framework for Mobile-Hybrid-Cloud Applications
  - Modeling of the Application Performance Maximization problem for Parallel Privacy-Preserving Hybrid Cloud offloading
  - Modeling of the Application Mobile Energy Consumption Minimization problem for Parallel Privacy-Preserving Hybrid Cloud offloading
  - An On-Line Monitor to profile dynamic system properties and to predict resource usage + Energy-model for fine-grained energy profiling
  - Policy-based offloading adjustments for Cost, Quality, Privacy
  - *Results show great potential for improving mobile-cloud application experiences for both developers and users*

OPEN SYSTEMS LABORATORY

The End: *An Adaptive Prog. Framework for Mobile-Cloud Apps*

A Flexible Fine-Grained Adaptive Framework for Parallel Mobile Hybrid Cloud Applications

Thanks for attending

Questions?

OPEN SYSTEMS LABORATORY