Tactics-Based Remote Execution

Rajesh Krishna Balan
Carnegie Mellon University

Thesis Committee
Mahadev Satyanarayanan
Greg Ganger
Srinivasan Seshan
David Garlan
Hari Balakrishnan

Motivation: mobile interactive applications
- speech recognition, language translation, augmented reality, …
  - Resource-heavy, but need bounded response time

Motivation: Handhelds are weak!
- Resource-intensive App
- Huge Data Sets

Motivation: But what about Moore’s Law
- Exponential resource increase
  - CPU, Memory, Disk
- Not quite true for handhelds
  - Battery capacity has increased only linearly
  - Wireless bandwidth not increasing as fast as wired bandwidth
    - 1 Gb/s versus 54 Mb/s
    - Very uncertain bandwidth
- Applications developed for desktops
  - Design/Space constraints
    - Memory, CPU sacrificed to obtain decent battery life in given form factor

Implications
- Handhelds will have problems running computationally intensive applications
  - Language translation, speech recognition, augmented reality etc.
- They can choose to
  - Not run these types of applications
  - Run these applications with terrible performance
  - Run specially created “lite” versions of these applications
    - Takes a long time to develop
    - Requires a separate development group
    - May not satisfy the user

Solution: Remote Execution
- Augment capabilities of handhelds by using nearby servers
- But how can good performance be achieved in mobile environments?
- And easily allow legacy applications to use remote execution?
- First work to address both concerns
Why is this Work Important?

Solution to a fundamental tension in developing mobile apps
- Mobile environments are inherently uncertain and dynamic
  » Bandwidth, availability of remote servers, battery life
  » Tackling this uncertainty requires automatic partitioning decisions
  » Otherwise, granularity of adaptation is too slow
- But, optimal partitions are frequently application and runtime specific
  » Automatic partitions may not perform well
- Tactics provide the required balance
  » Uses application/runtime specific information
  » Yet allows dynamic runtime adaptation to cope with uncertainty

Why is this Work Hard?

Formulating explicitly the notion of tactics
- What are their semantics?
- How do you specify them?
- What application types does it work for?
- What are the strengths and limitations of tactics?

Need to prove that tactics can be successfully used by an adaptive runtime system
- Runtime has to deal with the uncertain mobile environments
  » Yet make dynamic partitioning choices that are good
- Making existing applications use tactics must be easy
  » For system to be usable in practice
  » Achilles heel of many other adaptive systems

Roadmap

- Thesis Statement
  - Importance, Difficulty, Domain and Validation
- What are Tactics?
- Prototype Implementation (Chroma)
  - Components
  - Evaluation
- Tools for Supporting Tactics
  - Evaluation
- Timeline, Related Work & Conclusion

Thesis Statement

- Full range of meaningful partitions of an application can be described in a compact external description
  - Remote execution tactics
  - Partitions map remote execution possibilities of an app
- Demonstrate the two-fold benefits of tactics
  1. Enables powerful remote execution system
     » Provides good application performance at runtime
     » Low overhead
     » Automatic
  2. Amenable to sound software engineering principles
     » Decrease the time needed to develop mobile applications for new hardware platforms
     » “Easy” to use

Model of Applications

- Computationally intensive interactive applications have operations that do useful work
  » An "operation" is an application-specific notion of work
  » Translate a sentence
- 2 main flavours
  - Discrete
    » Language Translation
  - Continuous
    » Speech Recognition
    » Video Streaming
- Differ in 2 key aspects
  - Rate of input arrival
  - Cost of setting up state required to perform the operation

Model of Applications (Cont)

- Application is classified according to
  - It’s behaviour
    » Language Translation (Discrete)
      » 1 sentence every second
      » State cost is usually minimal
    » Speech Recognition
    » Video Streaming (Continuous)
      » 30 frames per second
      » Making a decision for each frame may take too long
      » Decoding frame x may be dependent on decoding frame x-1
  - Capabilities of current system
    » Slower system => fewer discrete applications
Issues Not Being Tackled

- Usability of handheld devices
  - Thesis focus is on achieving good application performance once the application is running
  - How the user interacts with the application is out of scope

- Security issues of using handheld servers
  - Minimal authentication of servers may be provided
  - No effort to tackle issues like
    » How do I ensure that the server does the correct thing
    » How do I prevent the server from reading my data

Proposed Validation

The following need to be validated

- Expressive power of tactics is adequate for the class of applications targeted
- Proof of concept of
  - Prototype tactics-based remote execution system
    » Can provide good application performance
  - Tools to assist application development
    » Can shorten application development time

Validation of Tactics

- Goal
  - Generality of tactics
    » Tactics semantics are able to support the class of applications targeted

- Validation via multiple application case studies
  - 4 discrete applications have currently been validated
  - At least 4 additional different applications needed for validation
    » To ensure that the application domain space is well mapped
    » Applications that I plan to add
    » Applications that I plan to add
- Map application
- Video streaming application

Validation of Remote Execution Prototype

- Goal
  - To show that tactics can be used to build high performance adaptive systems
    » Performance of prototype with various applications and environments is excellent

- Effectiveness in different resource environments needs to be measured
  - Ability of system to deal with resource uncertainty
    » Changes in wireless bandwidth
  - Addition or removal of remote servers
  - Initial testing with 3 applications has been promising

Validation of Tools

- Goal
  - Tactics are amenable to the use of tools and software engineering methods to ease application development time

- Validation of tools and methodology for easing application development time
  - Initial experiments with 4 applications have been positive
    » Method has to be applied to many more applications
  - When do the tools and methods help? When do they fail?
    » Compare time to add new applications with and without tools and methodology
    » By both experienced and inexperienced users
Current Status

Initial Tactics Model

Refine Tactics Semantics and Syntax

Runtime System

Software Engineering Components

Validation

Roadmap

- Thesis Statement
  - Importance, Difficulty, Domain and Validation
- What are Tactics?
- Prototype Implementation (Chroma)
  - Components
  - Evaluation
- Tools for Supporting Tactics
  - Evaluation
- Timeline, Related Work & Conclusion

Remote Execution Methods

- Static Partitioning
  - Easy to implement
  - Not flexible or effective
- Dynamic Partitioning
  - Most flexible and effective method
  - Extremely hard to implement
- Need a balance between the two

Solution: Tactics

- Concise description of application’s remote execution capabilities
  - Only the useful remote partitions are described
  - Can be captured in a compact declarative form
  - Allows use of stub generators to ease programming burden
- Tradeoff between dynamic and static partitioning
  - RPC model, no code migration
  - Coarse-grained remote execution

Tactic Semantics

- Tactics support the following semantics
  - Specify RPCs that make up the tactics
  - Allow RPCs to be executed in sequential order
  - Allow RPCs to be executed in parallel
    - Useful for applications with multiple optional components
    - E.g., language translation using different quality engines
    - Combination of sequential and parallel specifications
    - Specific server group specifications for any RPC
    - Useful for security or licensing reasons

Example Tactic

APPLICATION pangloss-lite;

/* RPC Specifications */
RPC server_dict (IN string line, OUT string dict_out);
RPC server_ebmt (IN string line, OUT string ebmt_out);
RPC server_lm (IN string gloss_out, IN string dict_out, IN string ebmt_out, OUT string translation);

/* Tactics (Useful Ways to Combine the RPCs) */
TACTIC dict = server_dict & server_lm;
TACTIC ebmt = server_ebmt & server_lm;
TACTIC dict_ebmt = (server_dict, server_ebmt) & server_lm;
Chroma

**Initial Validation**

- Four real research applications
  - Useful for mobile users
  1. **Pangloss-Lite** is a natural language translator [Federking @ CMU]
    - Valuable for travelers in foreign countries
  2. **Janus** is a speech recognizer [Waibel @ CMU]
    - Key component of speech interfaces
  3. **Faces** recognizes faces in images [Schneiderman @ CMU]
    - Representative of data mining applications
  4. **GLVU** renders 3D objects [Brooks @ UNC]
    - Augmented reality

**To Be Done**

- Validation of tactics
  - Currently, 4 applications have been looked at
  - More applications need to be described using tactics to map application domain
    » Good mix of continuous and discrete applications
- Refine syntax and semantics based on case studies

**Roadmap**

- **Thesis Statement**
  - Importance, Difficulty, Domain and Validation
- **What are Tactics?**
- **Prototype Implementation (Chroma)**
  - Components
  - Evaluation
- **Tools for Supporting Tactics**
  - Evaluation
- **Timeline, Related Work & Conclusion**

**Requirements of Prototype**

- Ability to understand tactics descriptions
- Aware of current resource availability
  - Number of remote servers
  - Available bandwidth
  - Battery
- Mechanisms to determine the best tactic for the available resources
  - Need to factor user preferences

**Components Needed**

- **Resource Monitors**
  - Battery, Bandwidth, CPU, Memory, Available Servers
  - Use existing service discovery protocols via middleware
- **Prediction of Resource Usage**
  - History Based Prediction (D. Narayanan’s work)
- **Solver to match the two**
  - Metrics for trading off resources
    » Latency
    » Battery Usage (J. Flinn’s work)
    » Fidelity
  - Influence of each metric in process is user-specific

**What is fidelity?**

- **Fidelity = runtime tunable quality**
  - Extent to which it matches a reference result
  - Used to compare quality of different executions of application
  - System changes fidelity by setting knobs in the application at runtime
- **Application-specific metric(s) / knob(s)**
  - resolution for augmented reality rendering
  - vocabulary size for speech recognition
  - JPEG compression level for web images
  - …
Do tactics allow good system performance?
- Performance of prototype for various applications will answer question

Performance of 3 initial applications measured
- Pangloss-Lite (Natural language translator)
- Janus (Speech recognizer)
- Face (Data mining)

Performance is comparable to hand-modified application
- Constant overhead added by runtime
- Overhead is low and reasonable

Problem: Making existing applications adaptive
- Modification of existing applications to support mobility is necessary
  - Not practical to rewrite existing apps from scratch
- However, these modifications are hard
  - Require expert knowledge of app
  - Need knowledge of runtime system
  - Modifications need to be updated every time either app or runtime changes

Roadmap
- Thesis Statement
  - Importance, Difficulty, Domain and Validation
- What are Tactics?
- Prototype Implementation (Chroma)
  - Components
  - Evaluation
- Tools for Supporting Tactics
  - Evaluation
- Timeline, Related Work & Conclusion

Tactics Don’t Hurt
Application: Janus, Metric = \[ \frac{\text{fidelity}}{\text{latency}} \]
Thinkpad 560X (200 Mhz Pentium)

What’s Next?
- Refine initial prototype
  - Support for continuous applications
  - Integration of service discovery middleware
  - Support for automatically using additional servers
    - Support for both famine and feast environments
- Integration with layer monitoring user preferences
  - To obtain hint modules
  - Prism in the Aura context
- Validation with numerous other applications
  - At least 2 continuous applications
Our Approach

- **Key insight**
  - Adaptation information can be described in an external form
  - Application and runtime independent
  - Cleanly separates adaptive behavior from application code
  - Leads naturally to code reuse etc.

- **Benefits of our approach**
  - Much easier to preserve adaptation across software changes
  - Separation and level of indirection allow tools like stub generators
  - Easier to change policies for adaptation
  - Allows development of a specialized software engineering methodology to ease development time

What Does It Work For?

- **Not a magic bullet**
- **Works for the following application types**
  - Well defined procedures for remote execution
    - Amenable to the RPC model
  - Well defined processing loop that performs the main work of the application
    - Rendering loop in graphics apps
    - Receive a sentence and translate it
  - Violation of either principle may require the application to be restructured
    - Costly process

API Calls (Current)

**Basic**
- Register: Register app with Chroma
- Cleanup: Destroys all data structures

**For Resource Logging**
- Start_operation: Tells Chroma to start logging resource usage
- Stop_operation: Tells Chroma when to stop

**Core Functionality**
- Find_fidelity: Ask Chroma to return appropriate fidelity vals
- Do_actics: Tells Chroma to perform any remote executions

Case Study: Pangloss-Lite

**Application panlite;**

```c
IN integer nwords FROM 0 TO infinity DEFAULT 1;
RPC server_gbt (IN string line, OUT string gbt_out);
RPC server_ebmt (IN string line, OUT string ebmt_out);
RPC server_lm (IN string gbt_out, IN string ebmt_out, OUT string translation);
TACTIC gbt = server_gbt & server_lm;
TACTIC ebmt = server_ebmt & server_lm;
TACTIC gbt_ebmt = (server_gbt, server_ebmt) & server_lm;
```

Case Study: Pangloss-Lite

Works for the following application types

- Well defined procedures for remote execution
  - Amenable to the RPC model
- Well defined processing loop that performs the main work of the application
  - Rendering loop in graphics apps
  - Receive a sentence and translate it
- Violation of either principle may require the application to be restructured
  - Costly process

**Application**

- **Source code**
  - Leads naturally to code reuse etc.
  - Separation and level of indirection allow tools like stub generators
  - Application and runtime independent
  - Adaptation => remote execution + changing runtime parameters

**Methodology to ease development time**

**What Does It Work For?**
“Comprehensive” Validation

- Method is somewhat language independent
  - Three test applications were all different languages
    - C, C++ and Ada
- Minimal hand modification of actual code
  - ~17 lines for Pangloss-Lite
  - ~50 lines for Face
  - ~25 lines for Janus

What’s Missing?

- Mechanisms to make server development easier
- Proper validation
  - Time to add applications to system using the tools and methods measured for
    - Experienced user
    - Novice user
  - Using a large number of applications (at least 3 more)
    - Mix of both discrete and continuous

Roadmap

- Thesis Statement
  - Importance, Difficulty, Domain and Validation
- What are Tactics?
- Prototype Implementation (Chroma)
  - Components
  - Evaluation
- Tools for Supporting Tactics
  - Evaluation
- Timeline, Related Work & Conclusion

Timeline

Related Work

Remote Execution Systems

- Application-Aware
  - Interested in obtaining good performance by modifying applications
  - Not meant for mobile environments
    - Abacus (Amiri), Coign (Hunt), Condor (Barnes),
  - Early work on adaptation for mobile environments
  - Required huge amounts of work to add applications to runtime
    - Odyssey (Noble, Narayanan, Fillon), Spectra (Filien),

- Application-independent
  - Concentrates on adapting applications with no application modifications
  - Does not perform as well as application-aware adaptation
    - Poppet (De Lara)

Related Work (Cont)

- Object Migration
  - Treats components as objects
  - Not meant for mobile environments
    - Emerald (Jul), Corba (Vinoski)

- Process Migration
  - Able to remotely execute arbitrary parts of an application
  - Difficult to use and not meant for mobile environments
    - Amoeba (Tanenbaum), Sprite (Ousterhout), Java RMI (SUN)

- Declarative Languages
  - More generic languages to solve a wider range of problems
    - 4GLs (Martin), Little languages (Bentley)
Related Work (Cont)
Software Engineering
- Reusing techniques in a new domain
  - Stub generators
    - RPC (Birrell, Nelson)
  - Methodology
    - Software Modularity (Parnas)

Expected Contributions
- Concept of tactics for abstracting remote partitions of an application
- Design of a tactics-based remote execution system
- Set of tools and methods to ease application development time
- Validation and demonstration using multiple real applications

Publications