Tactics-Based Remote Execution for Mobile Computing

Lei Nie
University of Minnesota - Twin cities
02/03/2015
Roadmap

- Motivation & Description of Tactics

- Prototype Implementation (Chroma)
  - Components

- Evaluation

- Related Work & Conclusion
Motivation: mobile interactive applications

- Language translation, speech recognition, face recognition, augmented reality, ...
- Resource-heavy, but need bounded response time
  » Unfortunately, handhelds are weak!! Why?
  - The optimal partitioning of an interactive application into local and remote components is highly application-specific and platform-specific.
  - Matters are made worse by the fact that mobile environments exhibit highly variable resource availability.
Solution: Remote Execution

- Augment capabilities of handhelds by using nearby servers
  - transform the puniest mobile device into a computing giant able to run resource-intensive applications

- But how can good performance be achieved in mobile environments?
- And easily allow legacy applications to use remote execution?
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
Key Insight

For a large number of applications

- Number of useful remote partitions is small
  - Largest so far is 7 partitions
    » Modular level coarse-grained partitions

- Application developer specifies these partitions (static partitioning)
  - At runtime, pick the optimal partition and locations
    (dynamic partitioning)
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
  - Each tactic performs the required operation

- Tradeoff between dynamic and static partitioning
  - RPC model
  - Assume servers have been discovered and are able to handle any RPC call (no code migration)
  - Coarse-grained remote execution
Example Tactic

Dictionary

Example based

Language modeler

RPC server_gloss (IN string line, OUT string gloss_out);
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 line, IN string ebmt_out,
                 IN string dict_out, IN string gloss_out,
                 OUT string translation);

DEFINE_TACTIC  gloss  = server_gloss & server_lm;
DEFINE_TACTIC  dict   = server_dict  & server_lm;
DEFINE_TACTIC  ebmt   = server_ebmt  & server_lm;
DEFINE_TACTIC  gloss_dict = (server_gloss, server_dict) & server_lm;
DEFINE_TACTIC  gloss_ebmt = (server_gloss, server_ebmt) & server_lm;
DEFINE_TACTIC  dict_ebmt  = (server_dict, server_ebmt) & server_lm;
DEFINE_TACTIC  gloss_dict_ebmt = (server_gloss, server_dict, server_ebmt) & server_lm;
Issues Not Being Tackled

- Usability of handheld devices
  - Focus is on achieving good application performance once the application is running
  - User interaction with the application is out of scope

- Security issues of using handheld servers
  - Minimal authentication of servers is 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
Roadmap

- Motivation & Description of Tactics
- Prototype Implementation (Chroma) - Components
- Evaluation
- Related Work & Conclusion
Requirements of Prototype

- Ability to understand tactics descriptions
- Aware of current resource availability
  - Number of remote servers
  - Available of remote servers
  - Battery
- Mechanisms to determine the best tactic for the available resources
  - Need to factor user preferences
    » Assume that this is provided by external entity (utility functions)
    latency-fidelity metric: fidelity/latency
Components Needed

- Resource Prediction
  - History Based Prediction (D. Narayanan’s work)

- Resource Monitors
  - Battery, Bandwidth, CPU, Memory, Available Servers
    » Use existing service discovery protocols via middleware

- User Guidance - Solver to match the two
  - Metrics for trading off resources
    » Battery Usage (J. Flinn’s work)
    » Fidelity
    » Latency
  - 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 – ...
Chroma

(Part of application) Description

Utility Function

User-specific knowledge

Inputs

Core functionality

Application

Tactics selection engine

Selected tactic

Operation executor

Resource availability

Predicted resource usage

Resource Monitors

Resource Demand Predictor

Log file

(tactic, resource consumption)
Chroma Goals

- Seamless from user perspective

- Effectiveness

- Minimal burden on application writers
  
  » How do we easily modify existing applications to use adaptation features provided by our runtime?

  - provide a lightweight semi-automatic process for customizing the API used by the application

  - provide tools for automatic generation of code stubs that map the customized API to the generic API used by Chroma

  - this generic API is supported by Chroma, which monitors resource levels and triggers application adaptation. Chroma support also helps ensure that the adaptations of multiple concurrently executing applications do not interfere with each other
Roadmap

- Motivation & Description of Tactics
- Prototype Implementation (Chroma)
  - Components
- Evaluation
- Related Work & Conclusion
Evaluation objective

- To show that Chroma has comparable performance to an oracle
  - Oracle selects best tactic for current environment
  - Oracle’s selection is determined offline while Chroma selects online

- Is the overhead of Chroma acceptable?

- Use of extra servers
  - What additional performance benefits are possible?
Applications Used

Three real research applications

- Useful for mobile users

1. **Pangloss-Lite** is a natural language translator
   [Federking @ CMU] (7 tactics)
   » Valuable for travelers in foreign countries
2. **Janus** is a speech recognizer [Waibel @ CMU] (2 tactics)
   » Key component of speech interfaces
3. **Face** detects faces in images [Schneiderman @ CMU] (1 tactic)
   » Representative of surveillance applications
Experimental Setup

- Thinkpad 560X Client (200 Mhz Pentium)
  - Representative of fastest handhelds

- HP Omnibook 6000 Servers (1 Ghz Pentium 3)

- 100 Mb/s Ethernet

Testing methodology
- Different inputs representing an operation
- Each result average of 20 runs
- State of system reset before each run
### Pangloss-Lite

<table>
<thead>
<tr>
<th>Sentence Length (No. of Words)</th>
<th>Ideal Runtime</th>
<th>Chroma</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>chosen tactic</td>
<td>metric</td>
<td>chosen tactic</td>
</tr>
<tr>
<td>11</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>gloss_dict_ebmt</td>
</tr>
<tr>
<td>23</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>dict_ebmt</td>
</tr>
<tr>
<td>35</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>dict_ebmt</td>
</tr>
<tr>
<td>47</td>
<td>gloss_dict_ebmt</td>
<td>0.70</td>
<td>dict_ebmt</td>
</tr>
<tr>
<td>59</td>
<td>dict_ebmt</td>
<td>0.70</td>
<td>dict_ebmt</td>
</tr>
</tbody>
</table>

(a) Fast Client

<table>
<thead>
<tr>
<th>Sentence Length (No. of Words)</th>
<th>Ideal Runtime</th>
<th>Chroma</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>chosen tactic</td>
<td>metric</td>
<td>chosen tactic</td>
</tr>
<tr>
<td>11</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>gloss_dict_ebmt</td>
</tr>
<tr>
<td>23</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>gloss_dict_ebmt</td>
</tr>
<tr>
<td>35</td>
<td>gloss_dict_ebmt</td>
<td>1.00</td>
<td>dict_ebmt</td>
</tr>
<tr>
<td>47</td>
<td>dict_ebmt</td>
<td>0.70</td>
<td>dict_ebmt</td>
</tr>
<tr>
<td>59</td>
<td>dict_ebmt</td>
<td>0.70</td>
<td>dict_ebmt</td>
</tr>
</tbody>
</table>

(b) Slow Client
Overhead Concerns

- Does the solver take too long?
  - Can it handle a large number of tactics?
  - What happens when the number of servers increases?

- Complete system overhead
  - How long does the system need to make decisions on a slow client?
    » Includes solver overhead + overhead of resource measurement & prediction
Overhead of Solver

- Synthetic results
- Slow client
- Max overhead < 0.9ms
- Okay for interactive apps

Graph showing time needed for solver (ms) vs. number of tactics for 20 servers, 10 servers, and 1 server.
Overhead of Entire System

Pangloss-Lite
Slow client
Max overhead < 1 s
Measures can be slow
- Caching helps
Adjusting to Over-Provisioned Environment

- Availability of compute servers varies wildly
  - Mobility ⇒ Cannot expect just one situation

- Tactics allow us to automatically use extra resources in environment
  - Without modifying the application

Resource Poor

Resource Rich

Smart Rooms etc.
Why is this useful?

- Fastest results
  - make multiple remote execution calls (for the same operation) to remote servers and use the fastest result

- Data decomposition
  - split the work necessary for an operation among multiple servers

- Best fidelity
  - perform the same operation but with different fidelities at different servers
  - return highest fidelity result that satisfies latency constraints
Tactics: Using Extra Servers

Still able to get performance improvements with loaded servers!!
Meeting Latency Constraints

<table>
<thead>
<tr>
<th>Application = Pangloss-Lite</th>
<th>Thinkpad 560X (200 Mhz Pentium)</th>
<th>Metric = fidelity/latency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fidelity</strong></td>
<td><strong>Latency</strong></td>
<td></td>
</tr>
<tr>
<td>Running to Completion</td>
<td>1.0</td>
<td>1.96</td>
</tr>
<tr>
<td>Taking Best Result after 1s</td>
<td>0.75</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Related work

- Application-aware remote execution systems
  - Abacus (Amiri), Coign (Hunt), Condor (Basney)
    » Not meant for mobile environments
  - Odyssey (Noble, Narayanan, Flinn), Spectra (Flinn)
    » Good performance in mobile environments
    » Hard to add new applications

- Other remote execution systems
  - Puppeteer (De Lara), Emerald (Jul), Sprite (Ousterhout)
Future Work

- Validate generality of tactics
  - Via multiple application case studies
  - Computationally intensive interactive apps

- Integration of services discovery mechanisms

- Use of extra servers
  - Distributed resource scheduling mechanisms
Conclusions

- Tactics are a valuable method of describing remote execution
- Tradeoff between static partitioning and full dynamic code migration
- Allows development of powerful remote execution systems
- Amenable to software engineering methods to ease application development
Thank you!