Tactics-Based Remote Execution for Mobile Computing

Rajesh Krishna Balan
Carnegie Mellon University

In Collaboration with
Mahadev Satyanarayanan (CMU)
SoYoung Park (CMU)
Tadashi Okoshi (CMU)

Motivation: mobile interactive applications
- speech recognition, language translation, augmented reality ...
  - Resource-heavy, but need bounded response time
    - Unfortunately, handhelds are weak!!

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?

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

APPLICATION pangloss-lite;

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;

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 bandwidth
  - 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)

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
  - …
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

Tactics Don’t Hurt

Application: Janus, Metric = \( \frac{\text{fidelity}}{\text{latency}} \)
**Tactics Don’t Hurt**

<table>
<thead>
<tr>
<th>Utterance</th>
<th>Oracle</th>
<th>Chroma</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency (s)</td>
<td>Fidelity</td>
<td>Latency (s)</td>
</tr>
<tr>
<td>1</td>
<td>0.71</td>
<td>0.50</td>
<td>0.73</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>0.76</td>
<td>0.50</td>
<td>0.80</td>
</tr>
<tr>
<td>4</td>
<td>0.78</td>
<td>0.50</td>
<td>0.79</td>
</tr>
<tr>
<td>5</td>
<td>0.90</td>
<td>0.50</td>
<td>0.93</td>
</tr>
<tr>
<td>6</td>
<td>0.95</td>
<td>0.50</td>
<td>0.98</td>
</tr>
<tr>
<td>7</td>
<td>0.70</td>
<td>0.50</td>
<td>0.71</td>
</tr>
<tr>
<td>8</td>
<td>1.00</td>
<td>0.50</td>
<td>1.22</td>
</tr>
<tr>
<td>9</td>
<td>0.77</td>
<td>0.50</td>
<td>0.77</td>
</tr>
<tr>
<td>10</td>
<td>0.99</td>
<td>0.50</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Application: Janus, Metric = \( \frac{\text{fidelity}}{\text{latency}} \)

Location: Remote in all case

**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 < 0.9 ms
- Okay for interactive apps

**Overhead of Entire System**

- Pangloss - Lite
- Slow client
- Max overhead < 1 s
- Measurers can be slow
  » Caching helps

**Why is this Useful?**

- Shields application from uncertainty in environment
  - Load on servers, wireless bandwidth
  - Execute same tactic on multiple servers
    » Take fastest result
- Opportunistic execution
  - To meet latency constraints
  - Execute multiple tactics on multiple servers
    » Return highest fidelity result that satisfies latency constraints

**Adjusting to 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.)

Chroma
Tactics: Using Extra Servers

Meeting Latency Constraints

Related Work

Future Work

Conclusions

Still able to get performance improvements with loaded servers!!

Application = Pangloss-Lite
Thinkpad 560X (200 Mhz Pentium)
Metric = \( \frac{\text{fidelity}}{\text{latency}} \)

Running to Completion

- Fidelity
- Latency
- Metric

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average (s)</th>
<th>Standard Deviation (s)</th>
</tr>
</thead>
<tbody>
<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>

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)

Validate generality of tactics

- Via multiple application case studies
- Computationally intensive interactive apps

Integration of service discovery mechanisms

- Use of extra servers
  - Distributed resource scheduling mechanisms

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