Mark S. Gordon, David Ke Hong, Peter M. Chen, Jason Flinn, Scott Mahlke and Zhuoqing Morley Mao
University of Michigan, Ann Arbor

TANGO: Accelerating Mobile Applications through Flip-Flop Replication

Mobile devices have less computational power and poorer Internet connections than other computers. Computation offload [1, 2, 3, 4], in which some portions of an application are migrated to a server, has been proposed as one way to remedy this deficiency. Yet, partition-based offload is challenging because it requires applications to accurately predict whether mobile or remote computation will be faster, and it requires that the computation be large enough to overcome the cost of shipping state to and from the server. Further, offload does not currently benefit network-intensive applications.

We propose an alternative approach for improving user-perceived performance on mobile devices. Our approach is based on replication rather than partitioning. Instead of predicting which parts of the code should run on the mobile device and which parts should run in the cloud, our system, called Tango, replicates the application and executes it on both the client and the server. Since either the client or the server execution may be faster during different phases of the application, Tango allows either replica to lead the execution and

Photo, istockphoto.com

attempts to reduce user-perceived application latency by predicting which replica will be faster and allowing it to lead execution and display output, leveraging the better network and computation resources of the server when the application can benefit from it.

Tango targets unmodified interactive applications that run in the Dalvik VM on the Android platform. Tango harnesses a trusted remote server to accelerate an application running on a mobile Android device. The server could be managed, for example, by a trusted cloud service provider and be located in a data center. Tango splits an application into a replicated portion and a non-replicated portion. The entire replicated portion runs on both the mobile computer and remote server. Tango uses techniques inspired by deterministic replay to keep the two replicas in sync, and it uses flip-flop replication to allow leadership to float between replicas. In contrast, different components of the non-replicated portion run on either the mobile computer or the remote server.

APPROACH

Tango uses five main techniques to improve user-perceived performance. First, Tango allows either replica to send output to the user. This is the fundamental property that reduces user-perceived latency, a Tango application will ideally appear to be running the faster replica — whichever replica that happens to be at the moment. For example, output from the remote server can be displayed on the mobile screen even before the mobile computation reaches the point in its execution where the output is produced.

Second, Tango uses deterministic replay to ensure the replicas perform the same computation and (importantly) produce the same output [5]. This guarantee is what lets Tango safely use the output of the first replica: the trailing replica will always produce the same result, so it does not matter which replica's output is externalized. Deterministic replay typically designates one replica as the leader. The leader logs all non-deterministic events (e.g., network input, user input and thread scheduling). The other replica is the follower — it supplies non-deterministic results from the log rather than re-executing such operations. Since the vast majority of operations performed by an application are deterministic, logging and replaying a (relatively) small amount of information ensures replica equivalency.

Third, in contrast to traditional deterministic replay, Tango shares the role of logging nondeterministic inputs from external sources between computers. Tango also shares the role of externalizing outputs from the replicated computation. For example, in the domain of mobile computing, some input sources are inherently tied to the mobile device (e.g., user input), while others are best tied to a remote server (e.g., network input, since the remote server usually has a better connection). Tango splits roles according to the type of I/O. The mobile device logs user input and sensor data, and it externalizes screen output. The remote server logs network input and externalizes network output.

Fourth, Tango replicates some I/O sources to reduce the amount of non-determinism that needs to be logged and to improve performance. Tango replicates data storage functionality, such as the file system and databases. It also replicates substantial portions of the user interface stack.

Fifth, Tango allows either replica to log internal sources of non-determinism, which include thread scheduling, time queries and asynchronous event scheduling. Unfortunately, prior uses of deterministic replay forced one computer (the follower) to always lag behind the other (the leader) to log such events. This a-priori designation limits performance in instances where the leader replica runs slower than the follower. Thus, Tango allows the role of leader to float between the two replicas: a technique we call flip-flop replication. When Tango predicts that the follower would be the faster replica, it flips the leader and follower roles. The role of leader may change many times during the execution of an application, e.g., due to alternation between periods of user interaction and periods of computation and/or network communication.

The Tango approach has many benefits. First, Tango achieves interactive performance that is very close to a system that always chooses the fastest location for execution. It does so without needing to predict resource availability, profile applications, or predict user behavior. Second, Tango supports low-overhead control transfer. In contrast to offloading, it does not need to ship all data used in offloaded computations because such data is automatically produced by the remote replica. This is particularly important because the amount of state reachable by offloaded computation can often be quite large. Third, Tango can provide fault tolerance by running multiple replicas. Importantly, by persisting its deterministic log within the cloud, Tango allows remote components to safely communicate over the network, so it hides the latency of multiple round trips over high-latency mobile networks. Finally, Tango can achieve these properties without modifying applications or profiling their executions.

WE PROPOSE AN ALTERNATIVE APPROACH FOR IMPROVING USER-PERCEIVED PERFORMANCE ON MOBILE DEVICES. OUR APPROACH IS BASED ON REPLICATION RATHER THAN PARTITIONING.

EXAMPLE SCENARIO

We illustrate how Tango works by describing how it would accelerate an example application. The scenario begins with the user interacting intensively with the application user interface. Events such as button and screen presses are broadcast to both replicas, but the client replica receives them first because communication with the server replica is subject to a network round trip. Thus, the client replica is the leader and decides when these events are scheduled into the application execution. The application may execute many synchronous native methods that query UI state; the leading client replica receives a quick response because of its co-location with the UI. In a standard thin

client solution, the server replica would lag far behind during this stage since it would initiate many synchronous UI requests and each would be subject to a full round-trip delay over a high-latency mobile network. In Tango, however, the UI events, the asynchronous scheduling decisions, and the result of synchronous native methods are all pushed to the server before the server asks for the results. Thus, all the information the server will need for its execution is sent in pipeline fashion, before the server even asks for the data, and the server replica lags only a one-way network delay behind the client replica at the end of this phase.

The application next enters a compute-intensive phase of execution triggered by the user's input (Figure 1a). Since the server has a faster processor than the client, the server replica quickly catches up and Tango switches leadership to the server. Note that because the server replica has followed the same execution as the client, it has the same state as the client replica and no state needs to be sent during the leader switch. During the compute-intensive phase of execution, the server replica calls UI methods that send output directly to the client native thread. The client displays this output even though the local replica on the mobile computer is lagging behind the server because it is still working through the computation that was just externalized by the client native thread. This allows the user to experience the faster responsiveness of the server replica during this phase. After receiving the output, the user spends some time thinking about his/her response to this output. While the user is thinking, the client replica catches up to the current state of the server.

When the user interacts with the UI, the client again becomes the leader. In the event that the client does not have time to catch up before the next user interaction, the server will continue to operate as leader until the client does catch up. During this time Tango will be operating like a thin client, with all user interactions being routed through the server.

The next phase of the application involves frequent network communication with an external computer (Figure 1b). The network operations are best done on the server, since it has a lower latency connection to the network. The server therefore assumes leadership. The network communication involves several rounds of synchronous message exchanges with the external computer. Each round involves sending a message to the external computer, waiting for its response, then computing the next step in the message exchange (e.g., an SSL handshake followed by several database queries with data dependencies). In an unmodified client, this phase is slow since it involves many round trips over the high-latency wireless network. However, the server replica interacts with the external computer much faster because it runs in a data center and has an excellent Internet connection. In a mirror image of the first phase, network input, asynchronous scheduling decisions from the leading server replica and responses to network methods are all sent to the client in pipeline fashion. Even with a slower processor, the client remains only a one-way network delay behind the server. Although the client takes longer to perform the small amount of computation in this phase, it catches up during network communication. The server waits for the remote computer to respond to each message, while the slightly-lagging client replica already has the response in its log. Thus, either the server or the client replica can quickly display output at the end of this phase. This multiphase execution may repeat indefinitely. For example, the application may next interact heavily with the user and so on. In each phase, Tango allows external components (i.e., the user and other computers) to experience the faster responsiveness of the replica that executes most quickly during that phase, while the slower replica can catch up to the faster replica during idle periods (e.g., when the application waits for the user and other computers). These two phases, compute-intensive and network-intensive, can be combined to get further benefits from Tango.

This scenario helps identify the specific instances in which we expect Tango to show

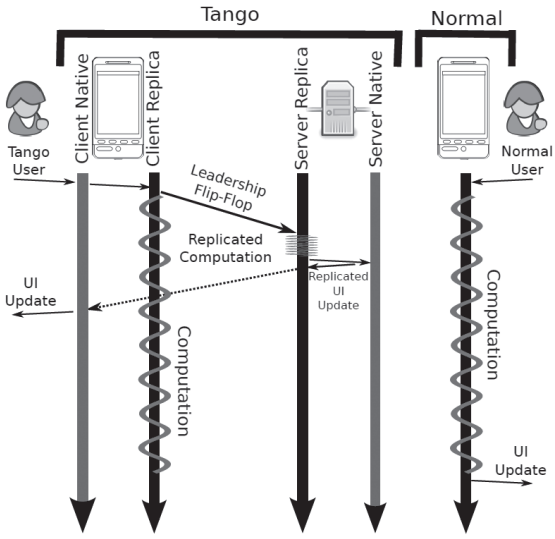

FIGURE 1(a). Compute-Intensive

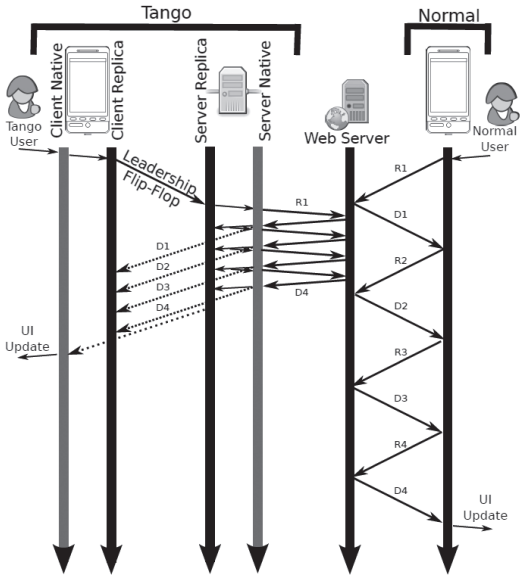

FIGURE 1(b). Network-Intensive

the most benefit. First, to benefit from use of a remote server, the application should include at least one computation-heavy phase or network phase with multiple communication round-trips. Second, for Tango to outperform a thin-client solution, the application should have a phase that interacts with the user or other I/O on the client. Finally, if the application has compute-heavy phases, there should also be phases with user think time or I/O during which the client computation can catch up with the server. We believe that many mobile applications share these exact characteristics, in other words they combine user interaction with computation and/or network communication.

EVALUATION

To provide support for a variety of mobile platforms, we implemented Tango on top of the CyanogenMod 10.1 code base. Its implementation details are discussed in [6]. We tested Tango on seven unmodified applications from Google Play to evaluate the performance benefits of Tango. Two of these applications, Sudoku and Poker, are compute-intensive, and the other five, Hoot, TapTu, E-mail, Instagram and Pinterest are network-intensive. Experiment results demonstrated that Tango achieves significant latency reduction with reasonable energy and communication overhead for both compute-intensive and network-intensive applications.

USER-PERCEIVED LATENCY

For each application, we configured Tango to connect its mobile client and replay server over USB for simulating typical network latencies and selected a representative latency-sensitive interaction to measure user-perceived latency. We recorded the time difference between the user input that triggered the interaction and the externalization of the final screen update as their user-perceived latency. Figure 2 shows the relative latency of Tango against the baseline for each application. For 100-ms round-trip network latency, which is the typical delay for current LTE networks, Tango reduced user-perceived latency for Sudoku by 0.6 seconds (50%) and Poker by 1.9 seconds (68%). Tango also achieved speedup of 1.3–2.6x under different network latencies for most network-intensive applications. Furthermore, the difference of speedup under 100ms and 500ms network latency indicates that the benefit of Tango increases with network latency for network-intensive applications and decreases for compute-intensive applications.

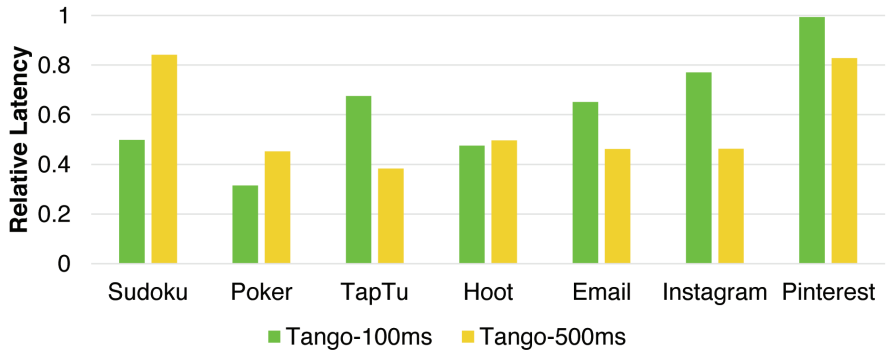

FIGURE 2. User-Perceived Latency

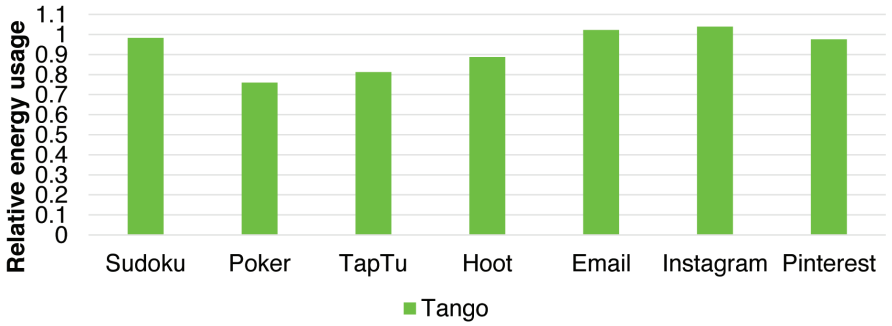

FIGURE 3. Energy Usage

ENERGY USAGE

We also measured the difference in energy usage caused by using Tango with each application in a WiFi network. We assume that a user will pause for a fixed think-time after seeing each screen update before beginning the next iteration of the benchmark. We used a user think-time of 3 seconds for Poker and 1 second for all other applications (in the case of Poker, this ensures that the client catches up during the designated user think time). Each energy measurement includes the time to finish all computation on both the client and server, as well as the fixed user think-time after displaying the result. Figure 3 shows the relative energy usage against the baseline for each application. Compared to the baseline, Tango uses less energy for most, but not all, benchmarks. ■

REFERENCES

[1] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang. The Case for Cyber Foraging. In the 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002.

[2] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. CloneCloud: Elastic Execution between Mobile Device and Cloud. In Proc. of the 6th ACM European Conference on Computer Systems, Salzburg, Austria, April 2011.

[3] E. Cuervo, A. Balasubramanian, D.ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making Smartphones Last Longer with Code Offload. In Proc. of the 8th International Conference on Mobile Systems, Applications and Services, pages 49-62, San Francisco, CA, June 2010.

[4] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen. Comet: Code Offload by Migrating Execution Transparently. In Proc. of the 10th Symposium on Operating Systems Design and Implementation, 2012.

[5] T. C. Bressoud and F. B. Schneider. Hypervisor-based Fault Tolerance. ACM Trans. on Computer Systems, 14(1):80–107, February 1996.

[6] M. S. Gordon, D. K. Hong, P. M. Chen, J. Flinn, S. Mahlke and Z. M. Mao. Accelerating Mobile Applications through Flip-Flop Replication. In Proc. of the 13th International Conference on Mobile Systems, Applications and Services, pages 137-150, Florence, Italy, May 2015.