Multiverse: Crowd Algorithms on Existing Interfaces

Kyle I. Murray
MIT CSAIL
Cambridge, MA 02139 USA
kimurray@mit.edu

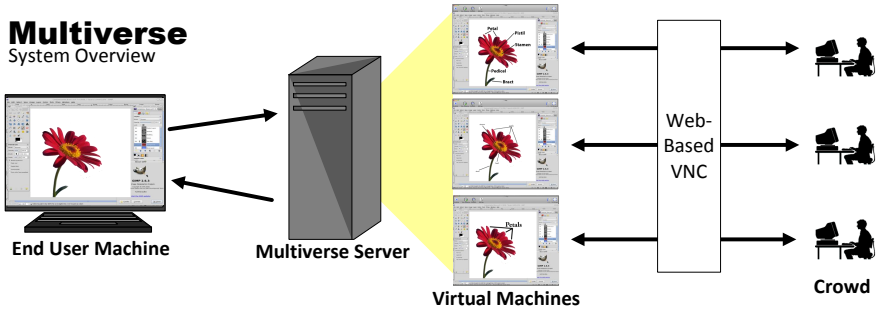

Figure 1: End users encapsulate tasks for the crowd in VMs that are then replicated on the Multiverse server and controlled by crowd workers via a web-based VNC connection. The server implements crowd algorithms on top of these VMs to ensure reliability.

CHI 2013 Extended Abstracts, April 27–May 2, 2013, Paris, France.
ACM 978-1-4503-1952-2/13/04.

Abstract
Crowd-powered systems implement *crowd algorithms* to improve crowd work through techniques like redundancy, iteration, and task decomposition. Existing approaches require substantial programming to package tasks for the crowd and apply crowd algorithms. We introduce Multiverse, a system that allows crowd algorithms to be applied to existing interfaces, reducing one-off programming effort and potentially allowing end users to directly employ crowdsourcing on the interfaces they care about. Multiverse encapsulates existing applications into cloneable virtual machines (VMs) that crowd workers control remotely. Because task state is captured in the VM, multiple workers can operate simultaneously on separate instances. We demonstrate the utility of this approach by implementing three existing crowd algorithms: (i) branch-and-vote, (ii) find-fix-verify, and (iii) partition-map-reduce. To implement these we introduce new crowd programming patterns: *crowd merge* and *crowd annotate*.

Author Keywords
Crowdsourcing; human computation; crowd algorithms

ACM Classification Keywords
H.5.2 [User Interfaces]: Graphical user interfaces.

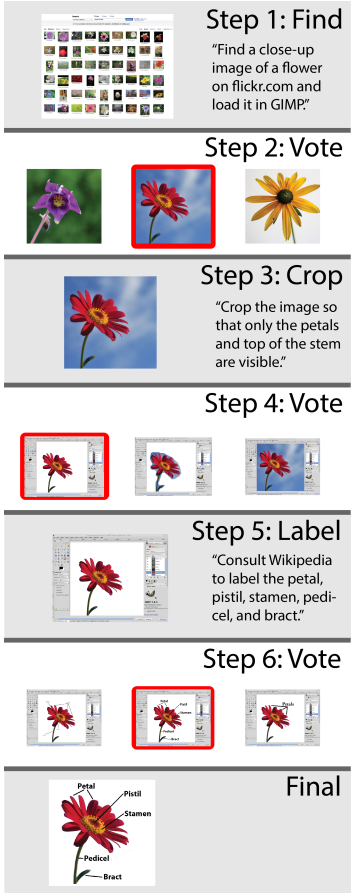

Figure 2: An example run of Multiverse. The task has been divided by the end user into three steps. Three workers perform each step separately in a private VM, and then other workers vote on the best, which is carried on to the next stage.

Introduction

Crowd-powered systems can now solve problems too difficult for automated approaches alone [3, 8, 4]. To leverage the potential of the crowd, these systems implement *crowd algorithms* that improve crowd work through techniques like redundancy, iteration, and task decomposition [3, 9, 11]. Existing approaches require substantial one-off programming effort to package tasks so that the crowd can work on them and so that crowd algorithms can be applied. In this paper, we introduce Multiverse, a system that allows crowd algorithms to be applied to existing interfaces, reducing one-off programming effort and potentially allowing end users to directly employ crowdsourcing on the tasks and interfaces they care about.

Multiverse encapsulates existing applications in virtual machine (VM) instances that can be copied and worked on separately by individual crowd workers, allowing crowd algorithms to operate on the existing application interfaces contained within the VM (Figure 2). To use Multiverse, an end user first decomposes her task into a sequence of steps and then initializes a VM into a good starting state (for instance, if they will be asking the crowd to do image editing, they might start up their image editor and load the image). Multiverse encapsulates this state into a VM file, copies the VM as appropriate, and allows multiple crowd workers to independently complete each sub-task through a Flash-based VNC connection to VM instances. This allows Multiverse to implement existing crowd algorithms; for instance, having multiple workers complete the same task and having a different set of workers vote on who completed the task best. The original end user can rejoin the VM when her task is complete.

The system is called Multiverse because it allows an operating system to exist in multiple states (or "universes") simultaneously through replication of VMs. The crowd creates and manipulates these universes and the end user gets the best one at the end. Multiverse is unable to apply crowd algorithms to steps that have side effects that escape the system, e.g. communicating with a remote server that stores its own data. This limitation is a result of how crowd algorithms work - using redundancy and layering to improve reliability.

To implement current crowd algorithms in Multiverse, manipulating VMs alone is not quite enough and so we have added two new crowd operations that broaden the kinds of algorithms that it can run. First, *crowd annotate* allows crowd workers to mark up existing interfaces via a visual drawing surface overlaying their view of the VM. For instance, for the find-fix-verify algorithm [3], crowd workers perform the find step by annotating interface in the overlay. The second operation is *crowd merge* which allows workers to merge portions of the states of multiple VM instances. This allows for merging (reduce step) in algorithms that use task decomposition, e.g. CrowdForge [8].

Our contributions are (i) introducing the idea of using VMs to enable crowd algorithms to be applied to existing interfaces and the Multiverse system that implements these ideas, (ii) implementing three common crowd algorithms on existing interfaces, (iii) and introducing the *crowd annotate* and *crowd merge* operations that make this possible.

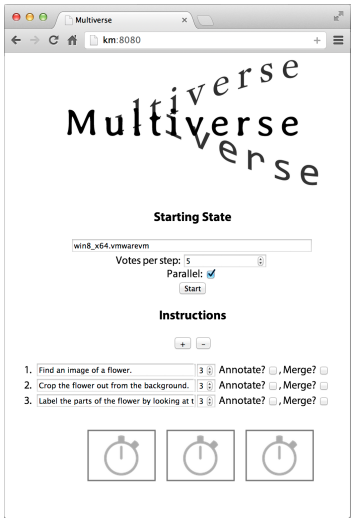

Figure 3: Multiverse users first prepare and save a VM in their OS of choice, and break their task into a sequence of steps. Multiverse recruits crowd workers to do these steps, and copies and manages connections to VM instances to make it work.

Background

Human computation was introduced to integrate people into computational processes to solve problems too difficult for computers, and has been shown useful in a variety of domains [3, 8, 4, 12, 6]. Soylent was the first to embed crowd work into an existing interface [3]. End users can access crowd work from within MS Word to help them complete tasks too difficult for automation - shortening, proofreading, and general human macros. Soylent required significant one-off programming effort and to appropriately package worker tasks so they could be completed on web-based crowd marketplaces like Mechanical Turk, whereas Multiverse allows crowd algorithms to be applied to any interface.

The Legion system allows crowds to control existing interfaces in real-time [11]. Legion workers collectively control a single interface instance in real-time. Legion trades accuracy for speed, and most existing crowd algorithms cannot be used in Legion because they are based on redundancy and iteration, which take time.

Multiverse is not the first system to copy and manage VMs, although usually this is usually done for reliability of the hosted platform [5]. The tutorial-based applications in T2A [10] allows a single user to explore variations in an existing application by running that application in multiple VMs. Its variations are limited to passing direct input from the end user to the VMs and automated GUI commands that were programmed for the system in advance. We believe Multiverse is the first system to allow the crowd to explore different courses of action within the same system.

Multiverse

Multiverse consists of 3 components: (i) server-side infastructure that clones and branches VM instances and manages crowd workers, (ii) an end user interface for spawning new multiverses, and (iii) a web-based crowd worker interface that connects workers to multiverse(s) via VNC (Figure 1).

Server Infrastructure

The server infrastructure manages VM instances and recruits crowd workers to complete tasks as needed using a custom Node.js [1] script. VMs are run in VMware Workstation [2] and coordination of the virtual machines is controlled by a custom Node.js script [1]. Multiverse can use any operating system that is supported by VMware Workstation, including most versions of Windows, Linux, and OS X.

End User Interface

The end user interface allows users to set a starting state, specify a crowd algorithm, and describe the task that is to be completed. Setting a starting state is as simple as operating the virtual machine. End users can also elect to simply start the virtual machine from a fixed initial state and instruct the crowd to initialize the state, e.g. “Start Powerpoint”, “Start Photoshop and open flower.jpg.” (Figure 3).

Crowd Worker Interface

The Multiverse interface for crowd workers displays instructions and one or more VNC interfaces for workers to control the remote virtual machine instances. VNC is provided by the Flash-based FlashLight-VNC client [7]. Specific instructions and the number and arrangement of the VNC clients are dependent on the crowd algorithm that is being used.

Crowd Algorithms on Existing Interfaces

In this section, we describe the three existing crowd algorithms that we implemented in Multiverse, the features of Multiverse used to support each, and the alternatives that could have been used. While it takes some programming effort to adapt a new crowd algorithm for use within Multiverse, once it has been adapted it can be applied to any interface.

Branch-Vote-Iterate

A common algorithm used in crowd-powered systems asks several workers to complete a task, then has other workers vote on which output was best, and then (optionally) forwards that output along to a next round. In Multiverse, this is achieved by branching VMs. The base VM is suspended so that several copies can be made. Each of these copies becomes a new branch for separate worker to pick up and work on. Critically, this suspension and copying technique for branching allows the applications within the VM branches to resume at the exact same point in execution where the base branch left off. These VMs can be run in parallel.

Find-Fix-Verify

In the find-fix-verify crowd algorithm, a set of workers finds problems to fix, a disjoint set fixes each problem, and then another disjoint set verifies that the fixes were correctly made [3]. When only sharing VMs, there is no consistent way for crowd workers to share problems that they find. Our solution to this is to overlay a drawing surface on top of the interface display to allow workers to mark up the interface. We call this *crowd annotate.* Workers recruited for the *find* stage are able to interact with the interface before they choose to start marking on it. Future versions of Multiverse may allow workers to switch multiple times between interactive and marking modes, and may attach annotations to specific portions of the interface using a pixel-based approach like Sikuli [13].

Divide-and-Merge

A common approach in crowd-powered systems is to break a large task into smaller subtasks, have crowd workers complete the smaller tasks, and then merge the results back together. Several systems have the crowd break the large task into subtasks [8, 9]. Multiverse provides a shared clipboard to support an operation that we call *crowd merge*. To support this, crowd workers are connected to two (or more) VMs. Workers can see and operate all of the VMs on the page as they run in parallel. A shared clipboard is a convenient mechanism to support merging across diverse applications because most applications already support storing their data on the clipboard. For instance, in PowerPoint one can copy and paste individual or groups of slides, drawing programs allow portions of the drawing to be copy and pasted, and spreadsheets allow cells to be copied while preserving underlying semantics. Multiverse implements its shared clipboard by displaying “copy” buttons next to all of the VMs except the first one, which is designated as the “paste” zone by convention. Activating the copy function in one VM will send a copy command to the virtualized operating system, serialize the contents of that machine’s clipboard, and finally send that data to the designated paste machine’s clipboard to await a “paste” command from the worker.

Evaluation

Experiments

The Multiverse server for all experiments was a six-core Intel Core i7 3.8GHz with 16 GB of memory

and a 512GB solid-state drive. This relatively powerful machine allowed us to run multiple virtual machines in parallel, and the SSD made copying rather large virtual machine files much faster. We initially conducted our experiments with crowd workers drawn from Amazon's Mechanical Turk, but we found that crowd workers on Mechanical Turk did not produce very good results (perhaps suggesting an avenue for future research), and so we instead ran our tests with a group of remote crowd workers recruited manually.

Tasks
We designed two multi-stepped tasks to illustrate the capabilities of Multiverse. The first shows Multiverse implementing the find-fix-verify algorithm. Workers were given a starting state of a simple mathematical diagram and asked to indicate the error in the diagram with the the marking tool. A voting step selected which of the workers' work from the first step would be used in the next. Workers were then instructed to fix the error in the diagram using the tools available in MS Paint.

The second task incorporated the merging feature to show how Multiverse can reduce the distributed work from a decomposed task back into a single state. Workers were each asked to design a slideshow slide about a particular themed food dish. In the merge step, the worker in charge was instructed to copy all of the slides into one virtual machine.

Discussion & Future Work

Multiverse suggests a number of potential future research opportunities. Multiverse potentially allows end users to use crowdsourcing in ways that they have not been easily able to do before. We have designed Multiverse to be easy to use – get a VM into a good starting state by operating it as you would any normal computer, choose an algorithm, and then list steps that you would like workers to do. Nevertheless, we plan to evaluate this with real end users to see both whether it is easy for them to package tasks in this way, if they find it useful, and what tasks they want to outsource in this way.

We have validated Multiverse through several examples. Future research may look how the design decisions made in developing Multiverse affects real-world performance. For instance, instead of showing screen shots to crowd workers in the `branch-vote-iterate` algorithm for voting, we could have shown a video or even allowed workers to interactively explore the VM.

Multiverse allows crowd algorithms to be applied to existing interfaces, but Multiverse may become even more useful as a component operating alongside other crowd-powered systems. First, applications that have side effects that are external to the VM would not be appropriate to control with Multiverse. Second, applications that need a response to the end user in real-time are not appropriate. An interesting direction for future work is to combine Multiverse with approaches that allow the crowd to do work in real-time, such as Legion [11]. A meta-level system could decide to use Multiverse to complete steps that do not have side effects and that do not need to be completed in real-time, whereas it could decide to use Legion to complete tasks that did. The crowd could even break down the task themselves so end users could provide a higher-level description [8, 9].

Multiverse may inspire new crowd algorithms. Just as

the find-fix-verify was initially motivated by text editing tasks, it may be that new crowd algorithms will be developed to handle types of tasks or applications that were inconvenient to explore before. Multiverse may serve as a platform for experts to provide assistance. For instance, users may pay an expert crowd worker to help them apply a difficult operation, e.g. a sequence of filters in Adobe Photoshop or a non-obvious setup step in an IDE.

Conclusion

In this paper, we have presented Multiverse, a system that allows end users to outsource jobs to the crowd on their existing applications without needing to repackage the request. We have shown how Multiverse can be used to apply crowd algorithms to existing interfaces, and introduced two new crowd operations, *crowd annotate* and *crowd merge*. Finally, we have demonstrated the utility of this framework by applying Multiverse on several realistic tasks.

Acknowledgements

We acknowledge advice and guidance from Jeffrey P. Bigham, Adam Marcus, and Robert C. Miller.

References

[1] Node.js, 2013. http://nodejs.org.

[2] Vmware workstation, 2013. http://www.vmware.com/products/workstation/.

[3] Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *UIST '10*, ACM (New York, NY, USA, 2010), 313–322.

[4] Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., and Yeh, T. Vizwiz: nearly real-time answers to visual questions. In *UIST '10*, ACM (New York, NY, USA, 2010), 333–342.

[5] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. Live migration of virtual machines. In *NSDI'05* (2005), 273–286.

[6] Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popovic, Z., and Players, F. Predicting protein structures with a multiplayer online game. *Nature 466*, 7307 (2010), 756–760.

[7] Fucci, M. Flashlight-vnc, 2012. http://flashlight-vnc.sourceforge.net.

[8] Kittur, A., Smus, B., and Kraut, R. Crowdforge: Crowdsourcing complex work. Tech. Rep. CMUHCII-11-100, Carnegie Mellon University, 2011.

[9] Kulkarni, A., Can, M., and Hartmann, B. Collaboratively crowdsourcing workflows with turkomatic. In *CSCW '12*, ACM (New York, NY, USA, 2012), 1003–1012.

[10] Laput, G., Adar, E., Dontcheva, M., and Li, W. Tutorial-based interfaces for cloud-enabled applications. In *UIST '12*, ACM (Cambridge, MA, USA, 2012), 113–122.

[11] Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *UIST '11* (2011), 23–32.

[12] von Ahn, L., and Dabbish, L. Labeling images with a computer game. In *CHI '04*, ACM (New York, NY, USA, 2004), 319–326.

[13] Yeh, T., Chang, T.-H., and Miller, R. C. Sikuli: using gui screenshots for search and automation. In *UIST '09*, ACM (New York, NY, USA, 2009), 183–192.