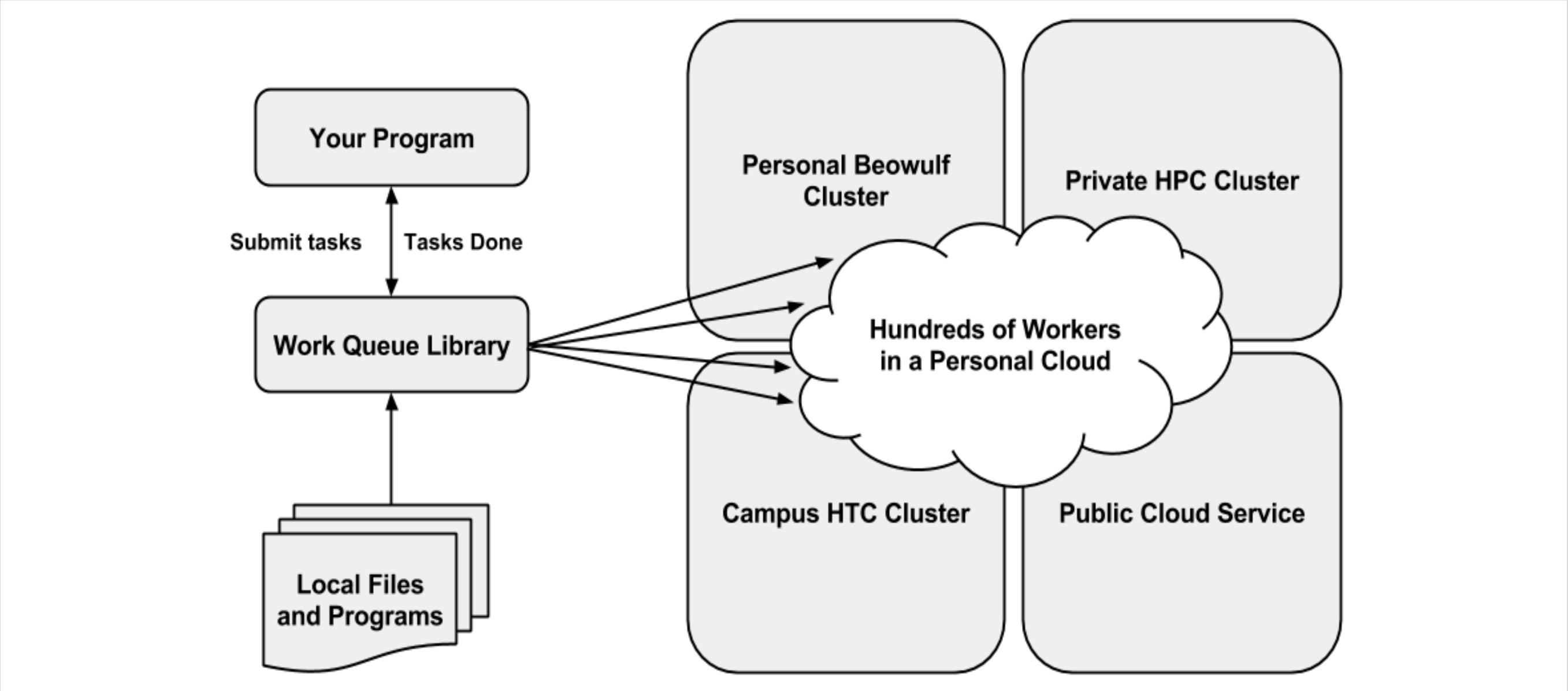
Work Queue: A Master/Worker Framework for PDC Education and Research

Peter Bui, Department of Computer Science, University of Wisconsin - Eau Claire

Work Queue

Work Queue [1] is a framework for building **large master-worker applications** that span many computers including **clusters**, **clouds**, and **grids**.

Work Queue applications are written in a variety of programming languages using a **simple API** that allows users to define tasks, submit them to the queue, and wait for completion.



Tasks are executed by a standard worker process that can run on any available machine. Each worker calls home to the master process, arranges for data transfer, and executes the tasks. The system handles a wide variety of failures, allowing for **dynamically scalable** and **robust** applications.

References

1. Peter Bui, Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, Douglas Thain. **Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications**. Workshop on Python for High Performance and Scientific Computing at SC 2011. November, 2011.
2. Nicholas Jaeger and Peter Bui. **To the Cloud and Back: A Distributed Photo Processing Pipeline**. Midwest Instruction and Computing Symposium (MICS2013). La Crosse, WI. April, 2013. Awarded Third Place.
3. Jeffrey Westphal and Peter Bui. **Scalable Distributed Image Transcoding using Python-WorkQueue**. Midwest Instruction and Computing Symposium (MICS2013). La Crosse, WI. April, 2013.
4. Peter Bui, Travis Boettcher, Nicholas Jaeger, Jeffrey Westphal. **Using Clusters in Undergraduate Research: Distributed Animation Rendering, Photo Processing, and Image Transcoding**. IEEE Cluster 2013. Indianapolis, IN. September, 2013.

Software

- **CCTools:** http://www.nd.edu/~ccl/
- **Work Queue MapReduce:** http://bitbucket.org/pbui/work-queue-mapreduce

Acknowledgements

We thank the following groups for supporting our work:

- **Cooperative Computing Lab**, University of Notre Dame
- **Office of Research and Sponsored Programs**, University of Wisconsin - Eau Claire
- **Center for High Throughput Computing**, University of Wisconsin - Madison

Education

Work Queue allows students to explore parallel and distributed computing in a variety of **courses** including electives, core classes, and introductory courses.

Cloud Computing

Explore MapReduce by analyzing Linux Source code

Using custom **MapReduce** implementation written in **Python** and **Work Queue** perform the following on the **Linux** source code repository:

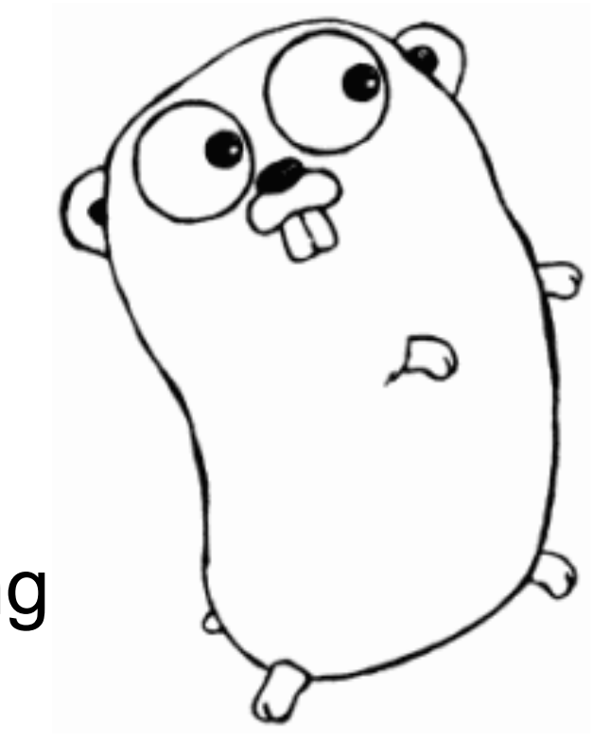
1. Generate **word frequencies**
2. Extract **email addresses** and **curse words**
3. Create **inverted index** of functions to source files
4. Determine **page rank** of each source file

Computer Organization and Design

Implement brute-force password cracker

Using the **Go** programming language and **Work Queue**, implement each of the following versions of a brute-force password cracker:

1. **Serial**
2. **Parallel** (multi-core)
3. **Distributed** (multi-node)



Once implemented, perform an **experiment** comparing all three versions.

Computing for the Sciences and Mathematics

Perform large scale data analysis and simulation

Using **IPython Notebook** and **Work Queue**, perform distributed simulations and analysis of various datasets:

1. **Aggregating and processing web data**
2. **Analyzing and filtering images**
3. **Monte Carlo simulations**

IP[y]: IPython Interactive Computing

Perform **visualization** of the data using the notebook interface.

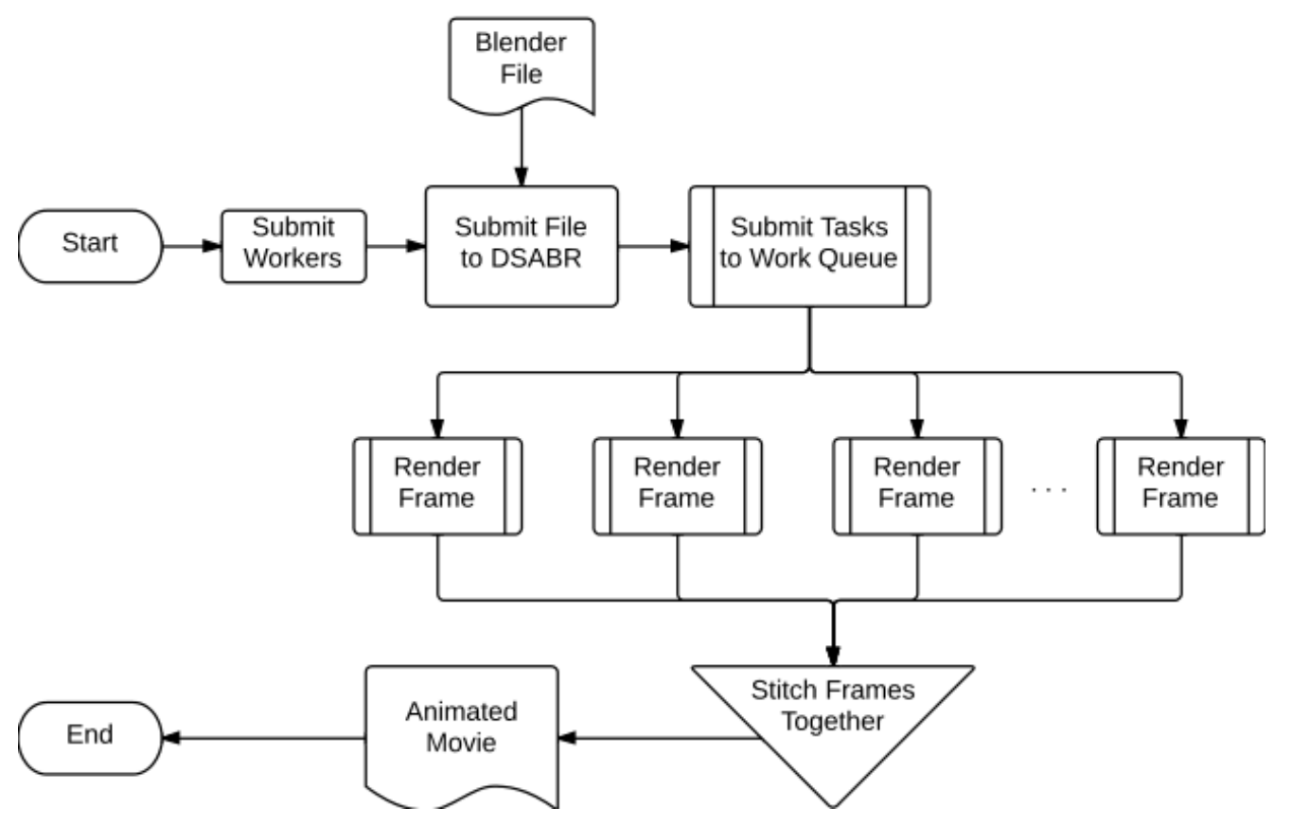
Research

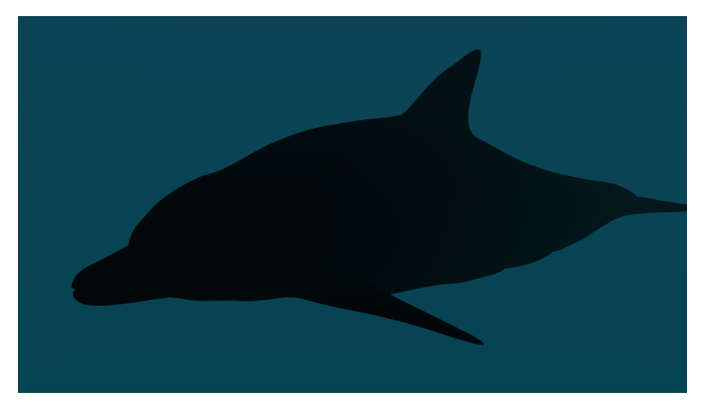
Work Queue enables undergraduates to quickly dive into **research** projects involving distributed computing **[4]**. It is particularly well-suited for data-intensive applications.

DSABR

Distributed system for automated Blender rendering

Using **Python** and **Work Queue**, implement a system for animating Blender files using multiple distribute clusters.

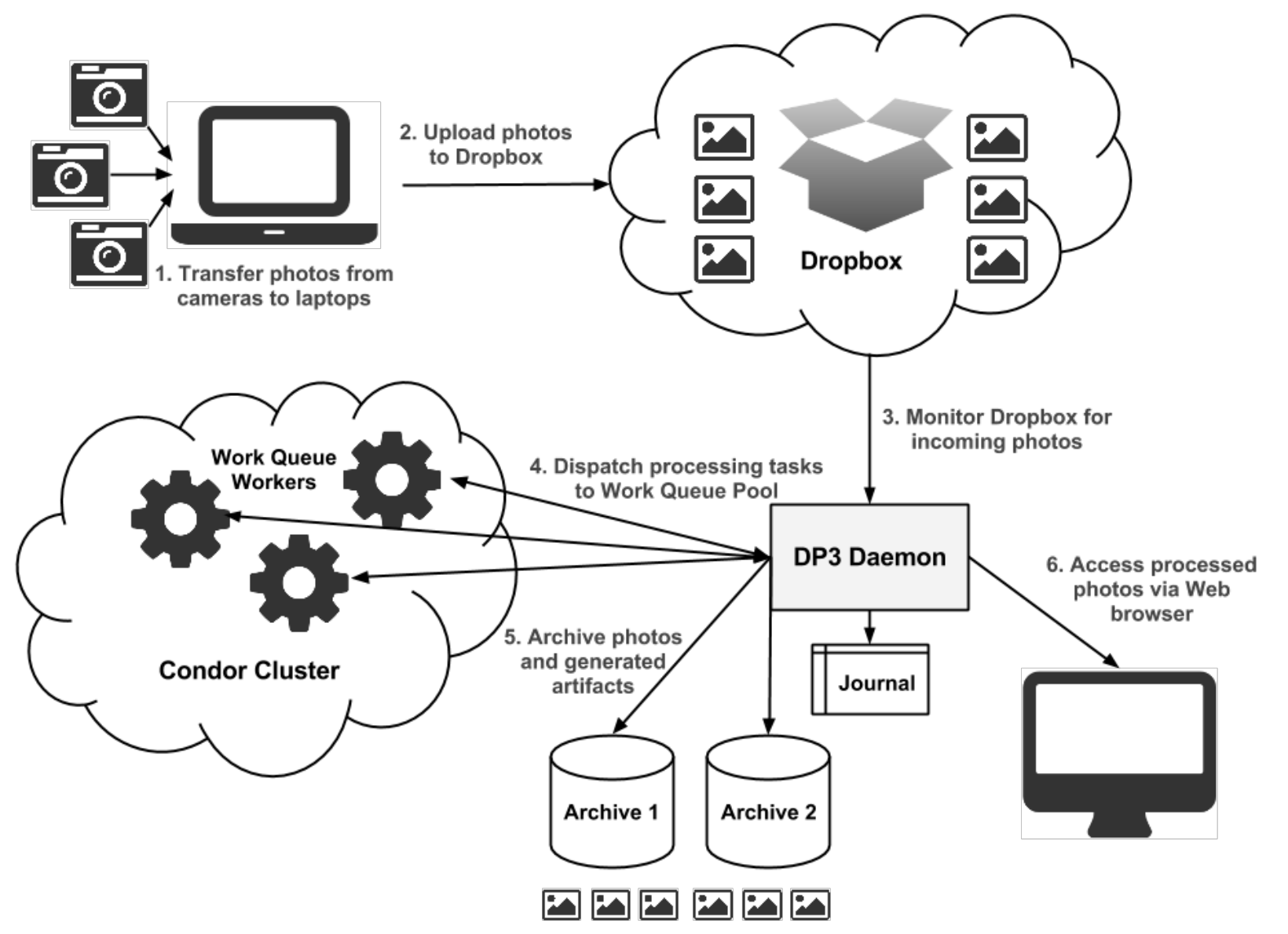




DP3

*Distributed photo processing pipeline **[2]***

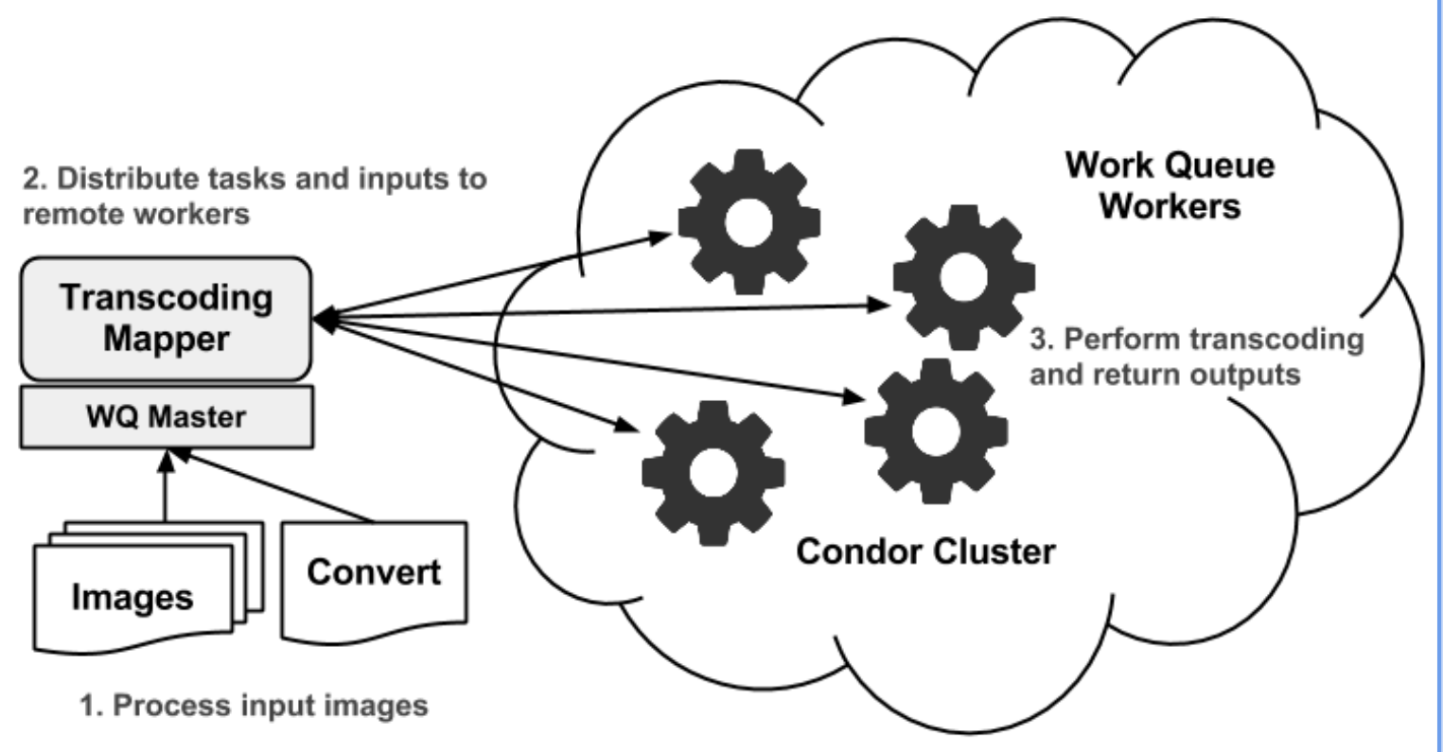
Support field scientists by allowing them to upload image data to a system that will **automatically process** and **archive** their **photos**.


1. Transfer photos from cameras to laptops
2. Upload photos to Dropbox
Dropbox
3. Monitor Dropbox for incoming photos
DP3 Daemon
4. Dispatch processing tasks to Work Queue Pool
Work Queue Workers
Condor Cluster
5. Archive photos and generated artifacts
Archive 1
Archive 2
Journal
6. Access processed photos via Web browser

Scalable Image Transcoding

*Test the scalability of an image processing workflow **[3]***

Use **Work Queue** to implement an **image processing** workflow that converts different sets of images into different formats and measure the application of the system.



File Size	Set Size	# of Workers 1	2	4	8	16	24	30
15KB	10	1x	1.47x	1.56x	2.13x	1.85x	2.00x	**2.40x**
	100	1x	1.60x	2.80x	4.43x	5.96x	6.42x	**6.44x**
	1000	1x	1.65x	3.12x	5.02x	7.97x	9.27x	**9.31x**
1MB	10	1x	1.65x	2.40x	2.78x	3.05x	3.73x	**3.87x**
	100	1x	2.10x	3.87x	6.55x	**9.56x**	7.65x	8.27x
	1000	1x	2.17x	4.28x	7.75x	11.2x	10.5x	**12.12x**
10MB	10	1x	1.84x	2.46x	2.88x	**4.48x**	3.43x	3.27x
	100	1x	1.98x	3.90x	4.95x	**7.34x**	4.61x	4.76x
	1000	1x	1.74x	3.97x	5.63x	**6.26x**	4.75x	4.93x