Distributed Web App Execution with Chunks

Justin Mazzola Paluska Hubert Pham Steve Ward
MIT CSAIL, Cambridge, MA, U.S.A.

1. Introduction Modern computer systems contain an enormous amount of structure (e.g., network fabrics, storage hierarchies, and software layers) that is hidden away from programmers by common frameworks (e.g., client/server or web programming). We explore a new computation model for dynamically organizing computation around the structure of available compute resources. In our model, all nodes share a unified execution environment and migrate code and computation between each other. Over time, a "client" node may (1) download more code to run locally to provide a richer user experience or (2) generate and upload code to remote servers to take advantage of extra processing power.

We demonstrate our model through PhotoBoss, a suite of web apps for editing high-resolution photos. PhotoBoss includes a photo organization tool, Quickr, and a photo editing tool, PhotoChop. PhotoChop exposes a number of customizable image filters to the user, many of which are compute-intensive and benefit from dynamic offload guided by the structure of available resources.

2. Chunk Abstraction Our previous work [1] explored a generic data model for low-latency access and fast manipulation of large objects that composes all objects as graphs of fixed-size, network-accessible *chunks*. A chunk is a fixed-sized array of fixed-sized *slots*. Each slot can contain either uninterpreted scalar data or an explicitly marked *link* to another chunk. Links contain references to, not addresses of, other chunks, allowing chunks to migrate among nodes sharing the chunks. Chunks abstract structure, enabling decision making based on what chunks link to which other chunks. Quickr uses chunks to expose multiple resolutions of each photo so that web clients may only download the particular resolution they need.

3. Computing with Chunks Our demo extends the data-only chunk model to include chunk-based computation within a chunk-based virtual machine called SCVM. SCVM maintains all of its runtime state in chunks, as shown in Figure 1. At the root of any computation is a `Thread` chunk that links to chunks containing all of the state for that thread of computation, including the program counter (PC), the environment (heap) of the chunk, and thread-local temporary storage. In order to expose the maximum amount of program structure in the chunk graph, we compile each basic block of code into its own chunk and use links to connect each basic block to other blocks to which it may continue, or to closures it may call.

PhotoChop runs an SCVM interpreter within the web browser to execute photo filter code. Since PhotoChop may run on a computationally weak web client, in order to

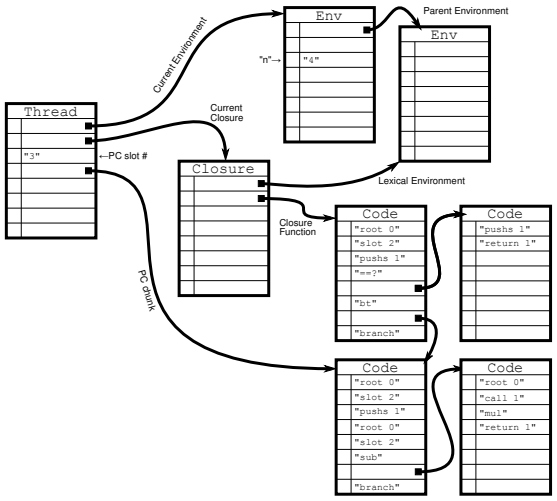

Figure 1: A SCVM thread running `factorial(4)`.

enable the user to interactively experiment with filter combinations and filter parameters, PhotoChop normally only operates on small resolution versions of photos. When the user finishes her experimentation, PhotoChop generates SCVM bytecode to apply the filters and usually migrates the code a remote server to operate on the high resolution photos. Depending on communication costs, PhotoChop may opt to run all of the bytecode locally, or split computation between local and remote servers.

To migrate computation between SCVM instances, the source VM need only send the root `Thread` chunk to the destination VM. As the destination VM reads links from the chunk, it will request more chunks from the source VM until it has copied enough chunks to make progress on the computation. While our migration strategy is similar to on-demand paging schemes, we may exploit our chunk representation to speed up migration by (1) sending only small portions of programs (such as individual functions or particular execution paths) rather than an entire binary and (2) using the structure encoded in the chunk graph to pre-fetch items [2].

4. Demo Requirements Our demo consists of PhotoBoss running on a laptop connecting to a remote server over a network. We will need power and a table, but can provide our own wireless network.

References

[1] Justin Mazzola Paluska and Hubert Pham. Interactive streaming of structured data. In *PerCom*, 2010.

[2] Justin Mazzola Paluska, Hubert Pham, and Steve Ward. Structuring the unstructured middle with chunk computing. In *HotOS*, 2011.