Sean Flinn - spf3d@uw.edu

Tarl Coufal - tarlc@uw.edu

Brian Oliphant

**Design**:

Thread Package -

Our sthread package has a few globa variables. It has an active thread which is currently running, a thread\_queue that has all ready threads on it, a dead\_thread\_queue that holds all dead threads who's internal elements can be freed, A return\_value\_queue to hold all threads that have finished running with threads waiting on them,

And an int (initted) to verify the package is initialized before being used.

The sthread struct holds the function and arguments it is going to run, sthread\_ctx\_t (stack and registers/thread state), whether the thread is joinable or not, a join\_queue where any thread's joining on that thread wait, a return value that can be quickly accessed by a joining thread if the thread has finished, and an int (is\_actually\_finished) to tell whether the thread has actually finished or not. We also use is\_actually\_finished to tell if thread is the main thread from spawned from init or another thread spawned from create.

We added a simple context switch method since context switches occurred so often, a runner function which acts as a wrapper function for threads so they have a place to return to (and exit cleanly), and a sthread\_user\_dispatcher function that simply calls yield() whenever a pre-emption interrupt occurs.

For pre-emption, we decided to disable interrupts any time there was a context switch or access to the thread\_queue. We decided to disable interrupts when accessing the thread\_queue, since locking seemed inefficient. If you lock the thread queue instead of turning off interrupts, a context switch cannot succeed, so interrupts won’t actually do anything useful.

Synchronization Primitives –

For our synchronization primitives, both the mutex and the condition variables have waiting queues. Whenever a user tries to get the mutex, they either succeed or are put on the waiting queue. At the same time, if the mutex is unlocked, it passes the mutex to the next thread on the waiting queue unless the queue is empty, then it is release.

The condition variable is very similar to the mutex, except it doesn’t get locked or unlocked. It simply adds threads to the waiting queue when they wait on the condition variable, adds one of the threads from the waiting queue to the thread\_queue when signal is called, and adds all of the waiting threads to the thread\_queue when broadcast is called.

Webserver –

The Webserver handles the requests by putting the connections on a queue with a max\_size (so you can’t keep overloading requests). It first creates the given input of worker threads which wait for the request queue to not be empty. Whenever the request queue isn’t empty, it gets a connection off the queue, then handles it. At the same time, if the queue ever gets full, the listener thread waits for the queue to no longer be full, so it can add more requests to the queue.

**Functionality:**

We believe everything is currently working

**Web Benchmarking:**

**Conclusions:**

This project was particularly challenging to think about because of the context switching. For example, even before working on preemption, we got confused on how sthread\_cond\_wait would release and reacquire the lock. I initially thought you would need to keep the locks on the queue with the waiting threads until I realized, you can just release the lock, put yourself on the queue, and then reacquire the lock in the wait method since the reacquiring of the lock won’t happen until after the thread gets switched back to.

It was also nice to learn more about the uses of threading, even when only one processor is being used at a time. Even without speeding up the program, it allows a user to divide the work in a more logical way.

One thing we did do that we could probably change is the use of the use of the return\_value\_queue. Since the return value is stored in the sthread, you should be able to access the return value without having to hold onto that thread in the thread package (assuming the user doesn’t free the thread prematurely).

Also, the context switch function is fairly useless. It was originally thought that yield would do more than just switch contexts, but in reality we could replace calls to the context switch function with yield and get the same results.