Data Races in GPU Programs

```
balance=10; balance+=10;
```

**Effects of Data Races**

Data races in the best case will crash your program. In the worst case they will silently lead to corrupt results. Gklee can detect such errors and produce traces to diagnose the cause.

**Race Detection Approaches**

- Conventional race checking tools use tests to expose programming errors. These tools are reliant on the quality of code coverage and may not expose bugs which depend on certain schedules.
- Symbolic tools, such as Gklee, can detect races with little user intervention. Gklee is also scheduler insensitive, as a canonical schedule can detect a race.

**Symbolic Execution Approaches**

- **Full Symbolic Execution**
  - Models all threads
  - Very slow when checking many threads
- **Parametric Execution (used by Gklee) with little user intervention.**
  - Take advantage of symmetry between threads modeling threads as "parametric flows"
  - Scheduler insensitive

**Disadvantage of Symbolic Execution**

- Slower than instrumentation based approaches
- Most time spent solving SMT queries
- Symbolic execution is done in series

**Potential Parallelization Approaches and Challenges**

We identified two approaches to adding parallel execution to Gklee:

- Execute parametric flows in parallel
  - Allows a fine time-slicing of the execution to be done in parallel
  - If no conflict Gklee re-checking race only within a flow, we get parallelized race checking as well
- Execute queries to the solver in parallel
  - Parallelize the most time consuming portion of execution
  - This is proved difficult to do with threads due to the structure of the used SMT solver, STP

**Parallelization Approach**

We chose to implement parallel parametric flow execution.

- Fork a process when flows fork
- Similar to the approach used by Cloud9[1]
- Races are checked per flow

**Results**

We measured a speed up of a factor of 2-8 in testing. The disparity is due to some forked flows exiting earlier than others.

**Future Work**

- We plan to add support for communicating changes in forked processes
- Implement better load balancing for parallel flow execution
- A solver server for parallelized race checking
- We are investigating using Swift/T to execute solver instances
  - Swift/T automatically load balancing and ensures dependencies are met before launching processes

**References**

Cloud9: A software testing service.