

Deconstructing Concurrency Heisenbugs

Shaz Qadeer

Research in Software Engineering
Microsoft Research

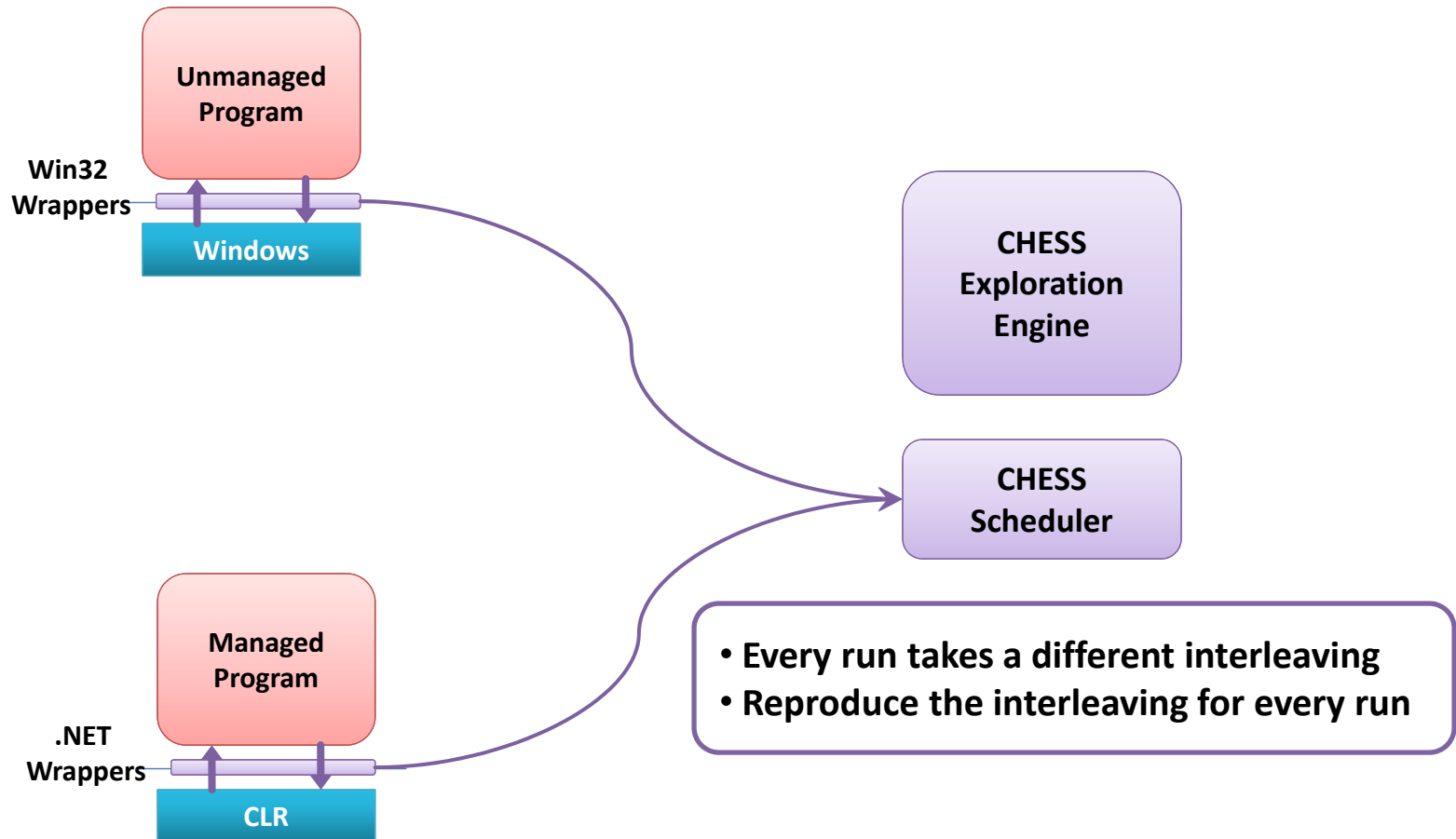
Concurrent Programming is HARD

- Concurrent executions are highly nondeterministic
- Rare thread interleavings result in Heisenbugs
 - Difficult to find, reproduce, and debug
- Observing the bug can “fix” it
 - Adding a print statement can change the scheduling behavior
- A huge productivity problem
 - Developers and testers can spend weeks chasing a single Heisenbug

CHESS in a nutshell

- CHESS is a user-mode scheduler
- Controls all scheduling nondeterminism
 - Replace the OS scheduler
- Guarantees:
 - Every program run takes a different thread interleaving
 - Reproduce the interleaving for every run

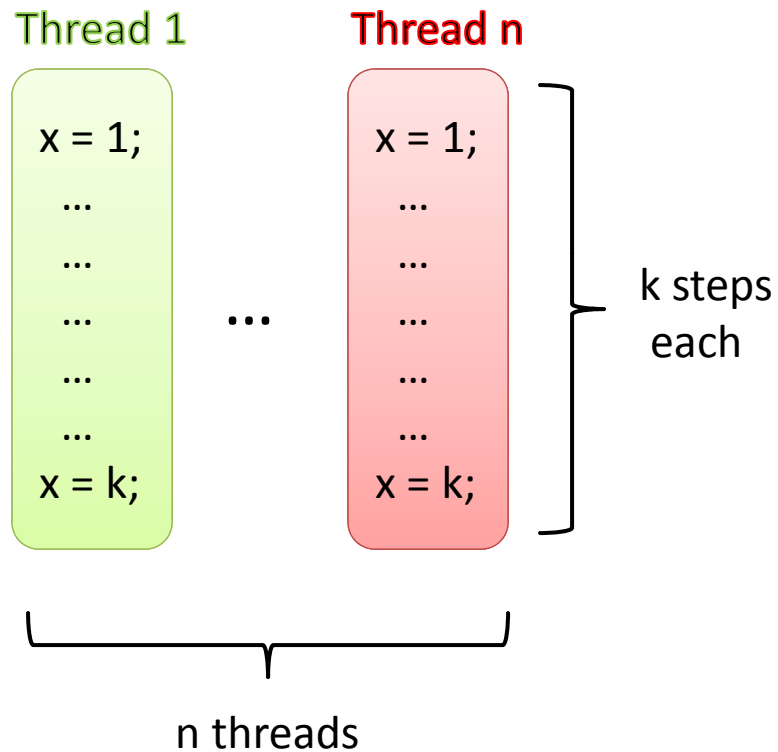
CHES architecture



Errors that CHESSE can find

- Assertions in the code
- Any dynamic monitor that you run
 - Memory leaks, double-free detector, ...
- Deadlocks
 - Program enters a state where no thread is enabled
- Livelocks
 - Program runs for a long time without making progress
- Dataraces
- Memory model races

State space explosion



- Number of executions
= $O(n^{nk})$
- Exponential in both n and k
 - Typically: $n < 10$ $k > 100$
- Limits scalability to large programs

Goal: Scale CHES to large programs (large k)

Preemption bounding

- By default, CHES is a non-preemptive starvation-free scheduler
 - Execute large chunks of code atomically
- Systematically insert a small number **preemptions**
 - Preemptions are context switches forced by the scheduler
 - e.g. Time-slice expiration
 - Non-preemptions – a thread voluntarily yields
 - e.g. Blocking on an unavailable lock, thread end
- Most errors are caused by few (≤ 2) preemptions

Polynomial state space

- Terminating program with fixed inputs and deterministic threads
 - n threads, k steps each, c preemptions
- Number of executions $\leq \binom{n+k}{k} \cdot (n+k)!$
 $= O((n^2k)^c \cdot n!)$

Exponential in n and c, **but not in k**

Progress report

- CHESSE used by Microsoft product groups
 - Parallel Computing Platform (PCP)
 - SQL
 - Windows CE
 - Midori
- External release via DevLabs
 - <http://msdn.microsoft.com/devlabs/>
- Academic release
 - <http://research.microsoft.com/en-us/projects/chess/>

Goal: Enable principled concurrent programming (I)

- Uncontrollable nondeterminism is the fundamental problem
- Two options
 - Deterministic semantics
 - Runtime hooks to expose and control nondeterminism
- Remember that sequential programming works primarily because the programmer can control and examine the computation

Goal: Enable principled concurrent programming (II)

- Compositional methods for reasoning
 - Formal or informal
- For sequential programs, we have
 - Stack abstraction (pre and post conditions)
 - Data abstraction (invariants)
- What are the appropriate abstractions for concurrent programs?