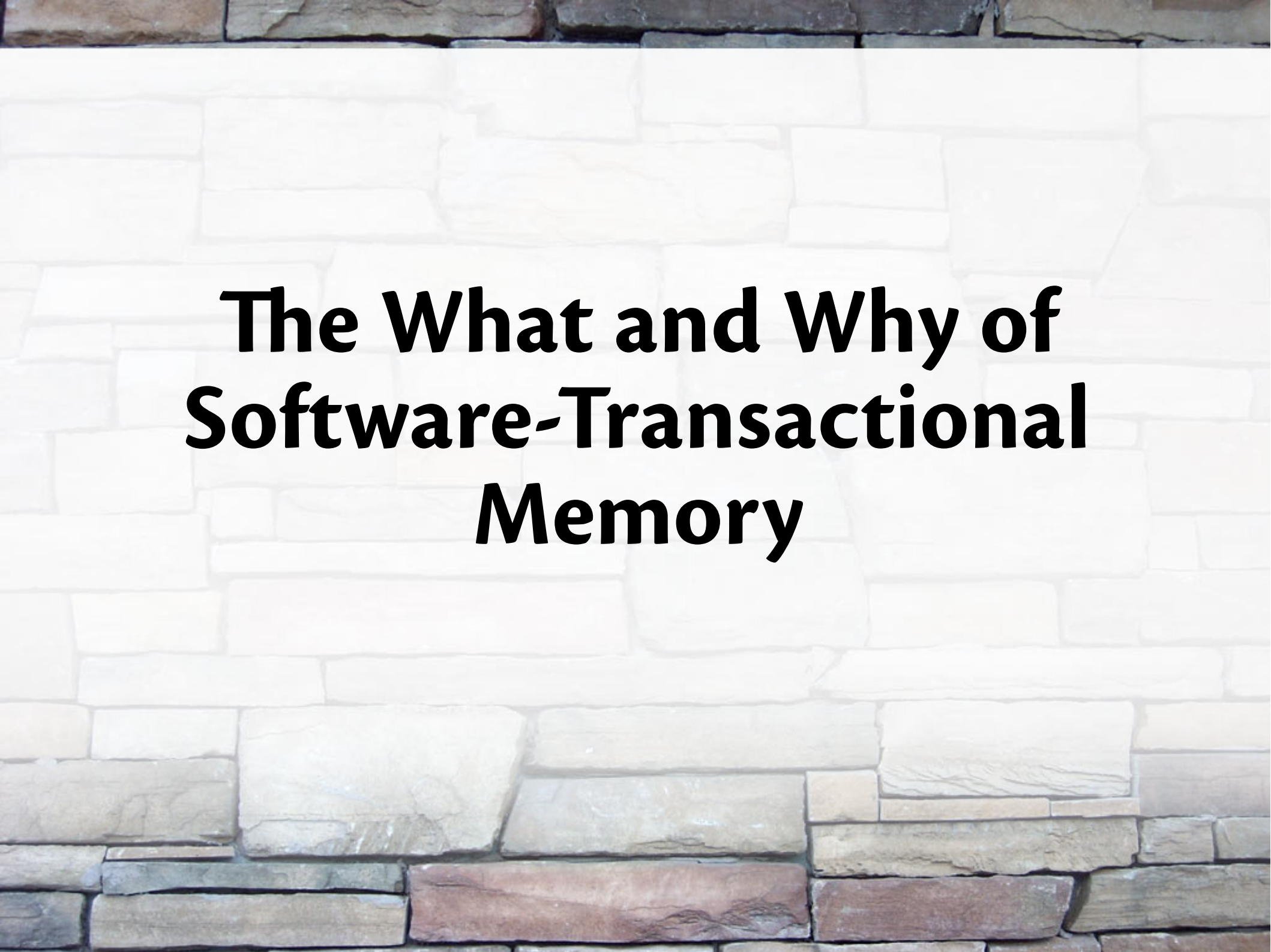
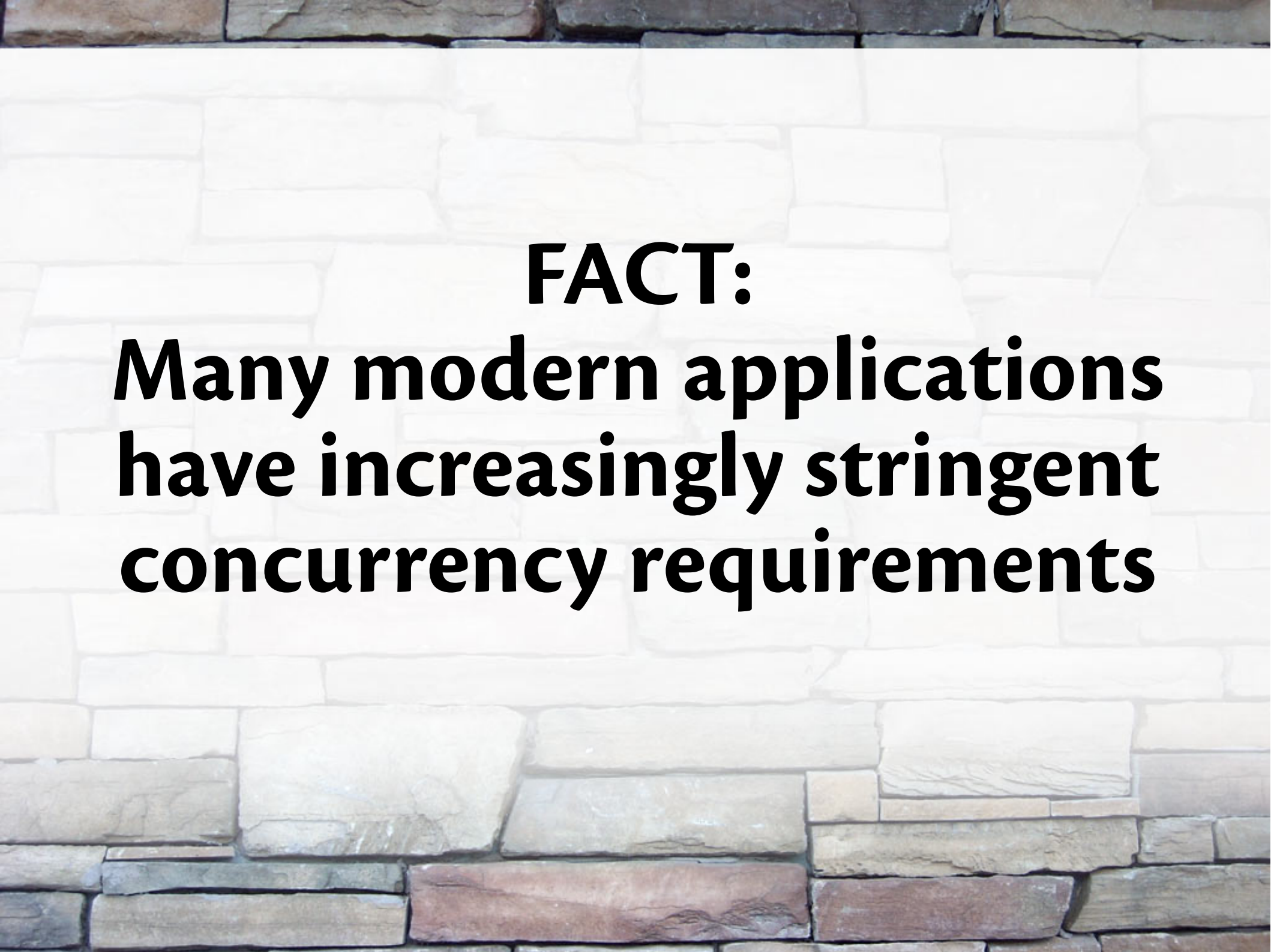
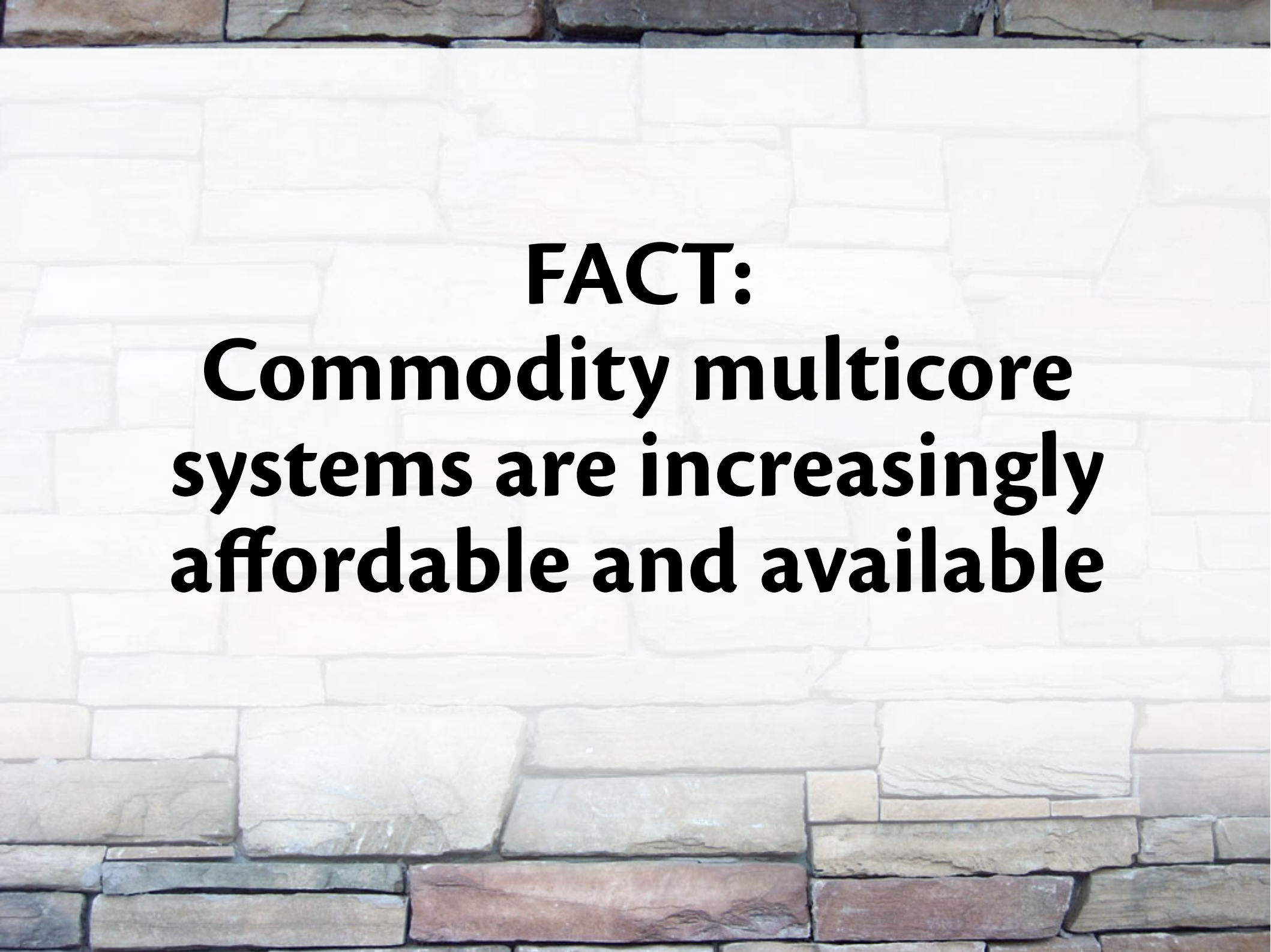
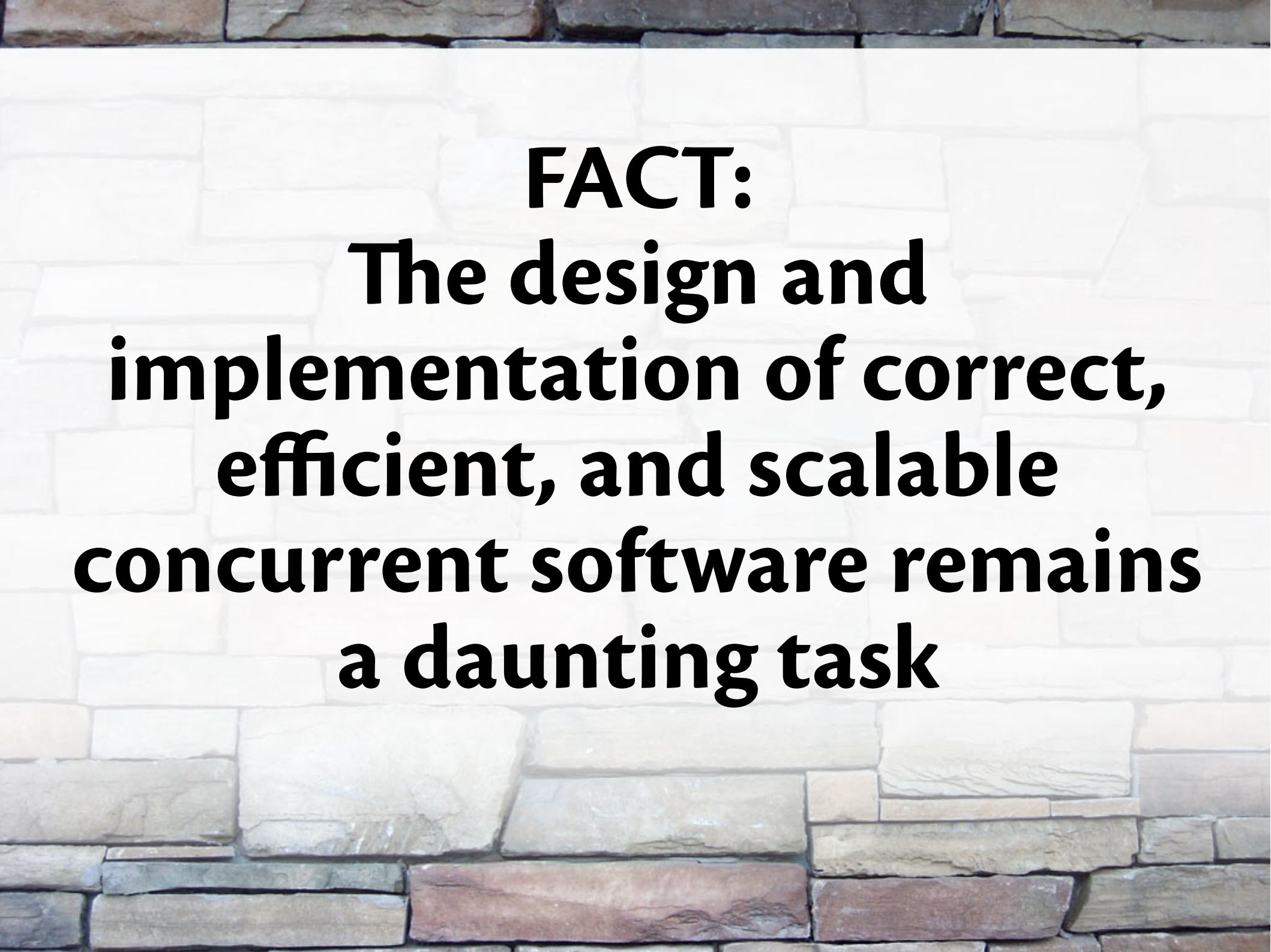

The What and Why of
Software-Transactional
Memory

Let's start with WHY

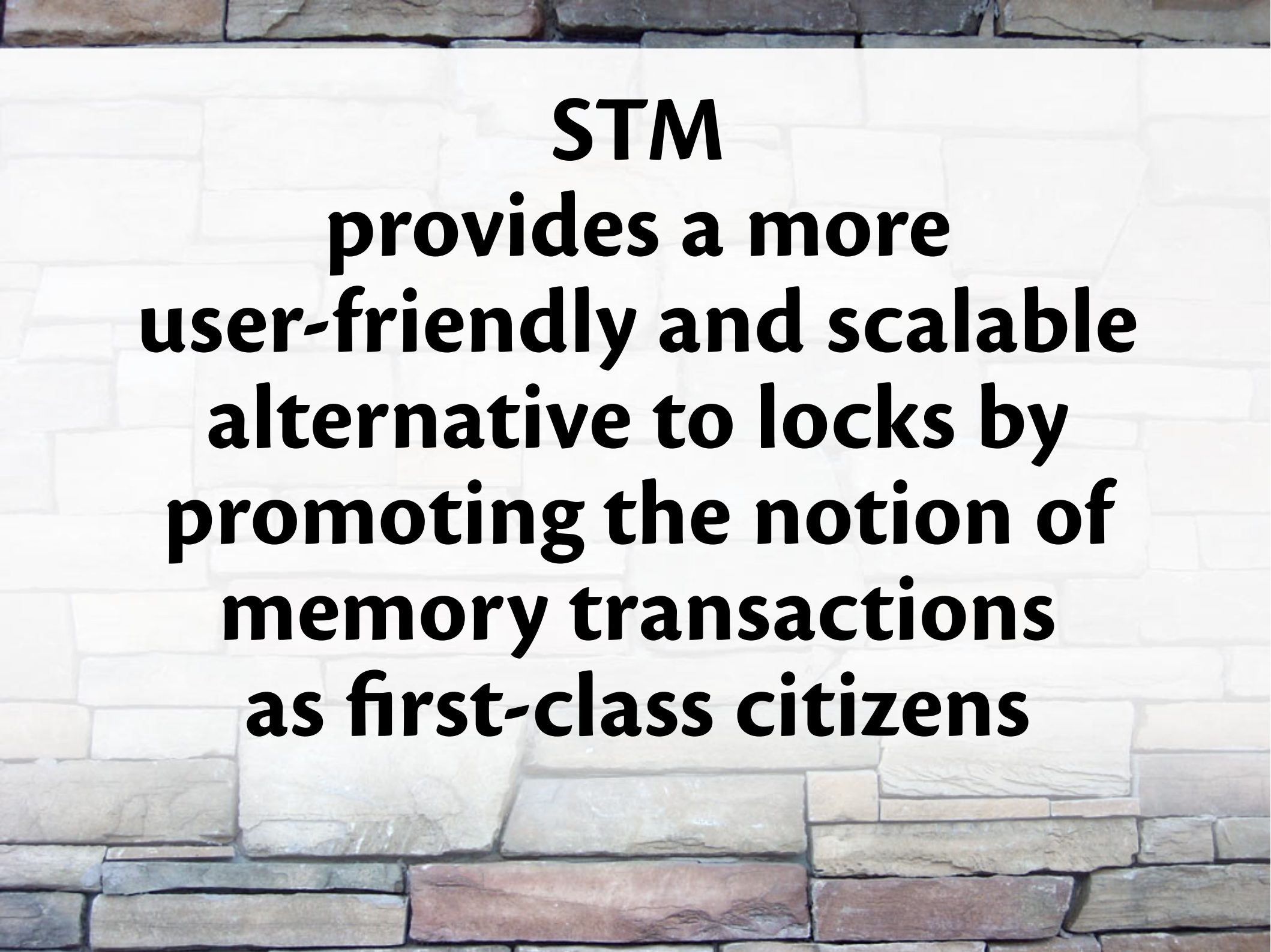

FACT:
Many modern applications have increasingly stringent concurrency requirements

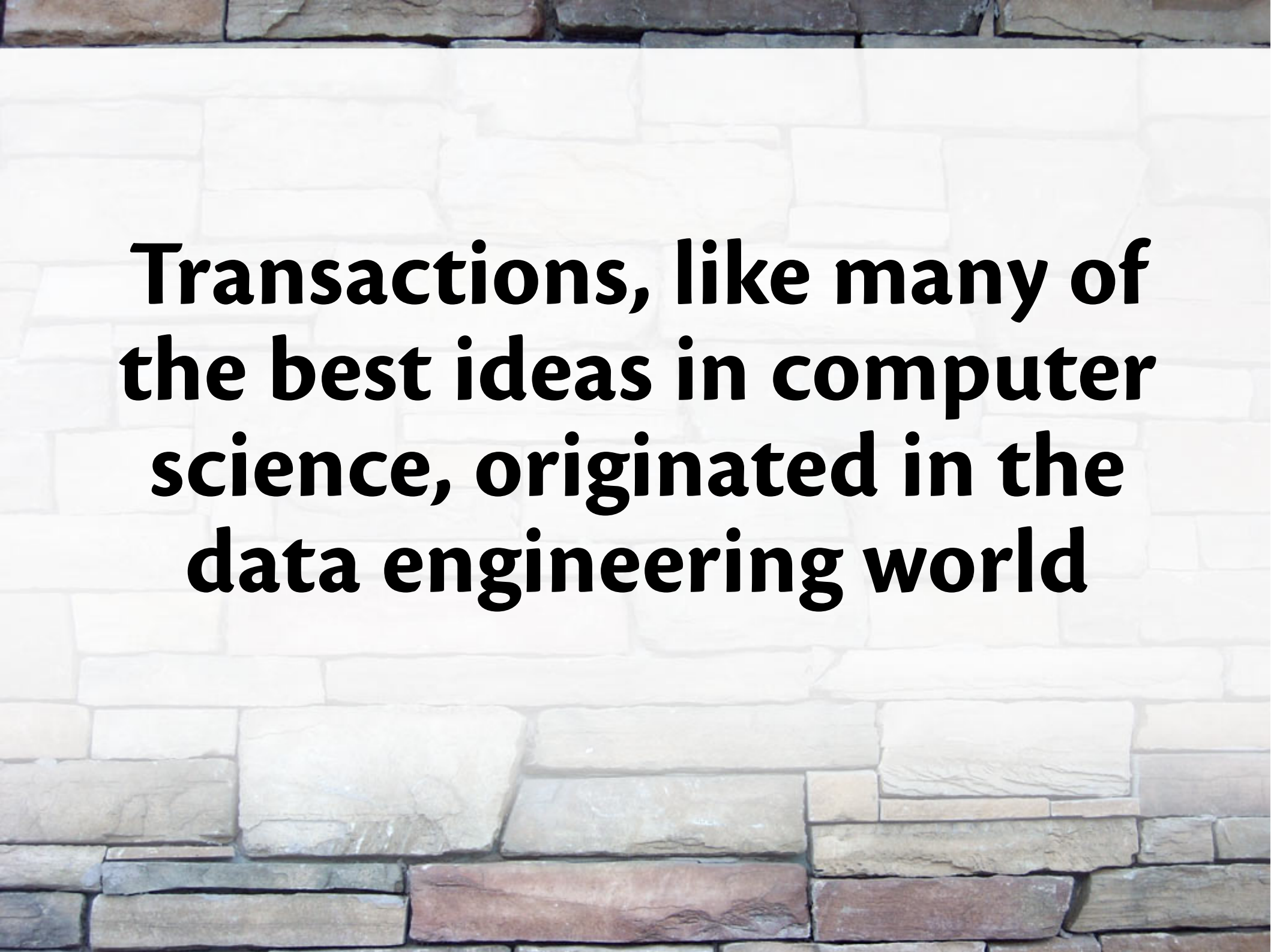

FACT:
Commodity multicore systems are increasingly affordable and available


FACT:
The design and implementation of correct, efficient, and scalable concurrent software remains a daunting task

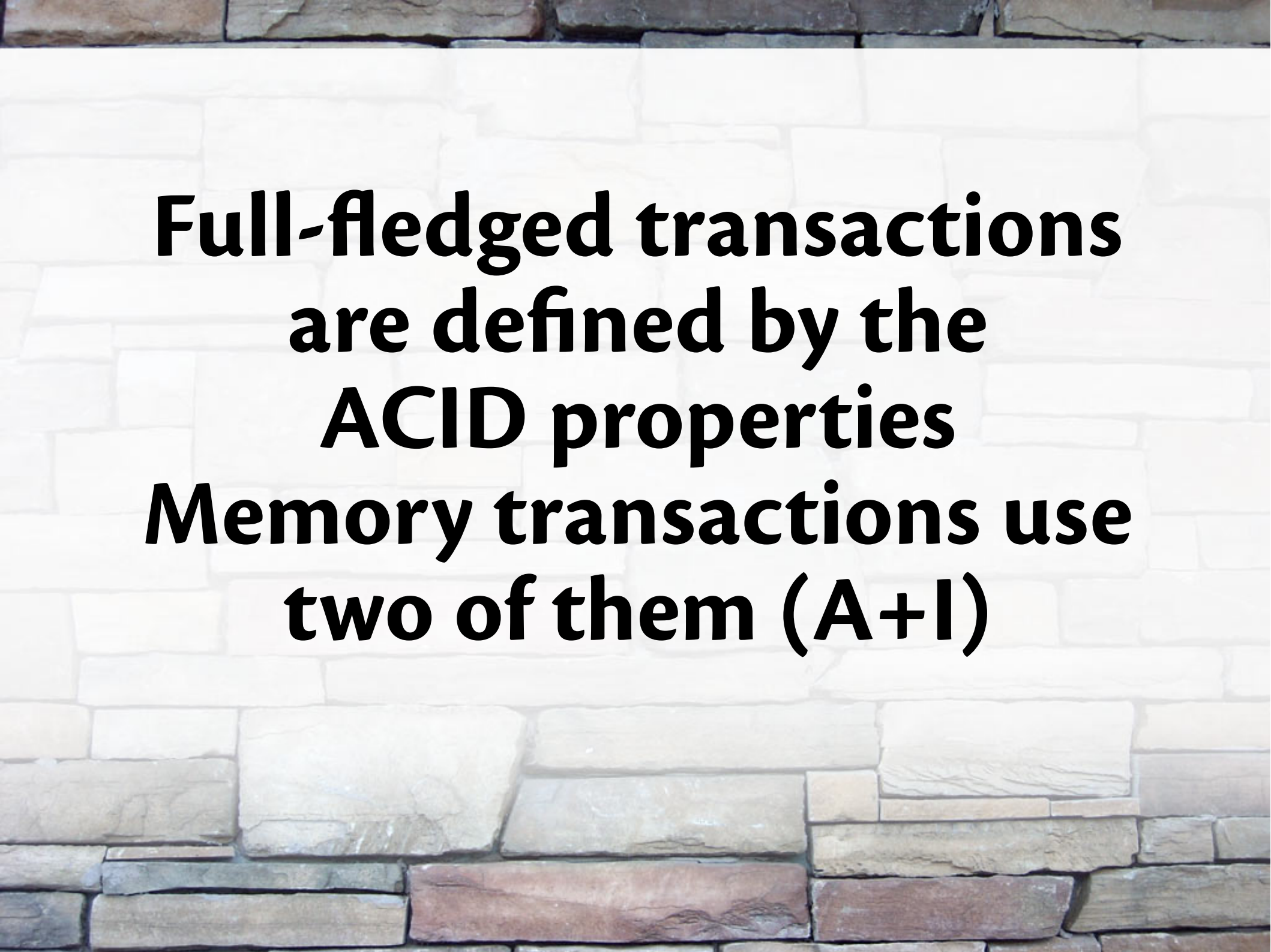

Haskell to the rescue!
Meet STM


STM
protects shared state in
concurrent programs


STM
provides a more
user-friendly and scalable
alternative to locks by
promoting the notion of
memory transactions
as first-class citizens


Transactions, like many of the best ideas in computer science, originated in the data engineering world


Transactions
are one of the foundations of
database technology


Full-fledged transactions
are defined by the
ACID properties
Memory transactions use
two of them (A+I)

Transactions provide atomicity and isolation guarantees

(SQLite transactions demo)

Strong atomicity means all-or-nothing


Strong isolation
means freedom from
interference by other
threads

Recall that Haskell is a strictly-typed, lazy, pure functional language

Pure means that functions with side-effects must be marked as such

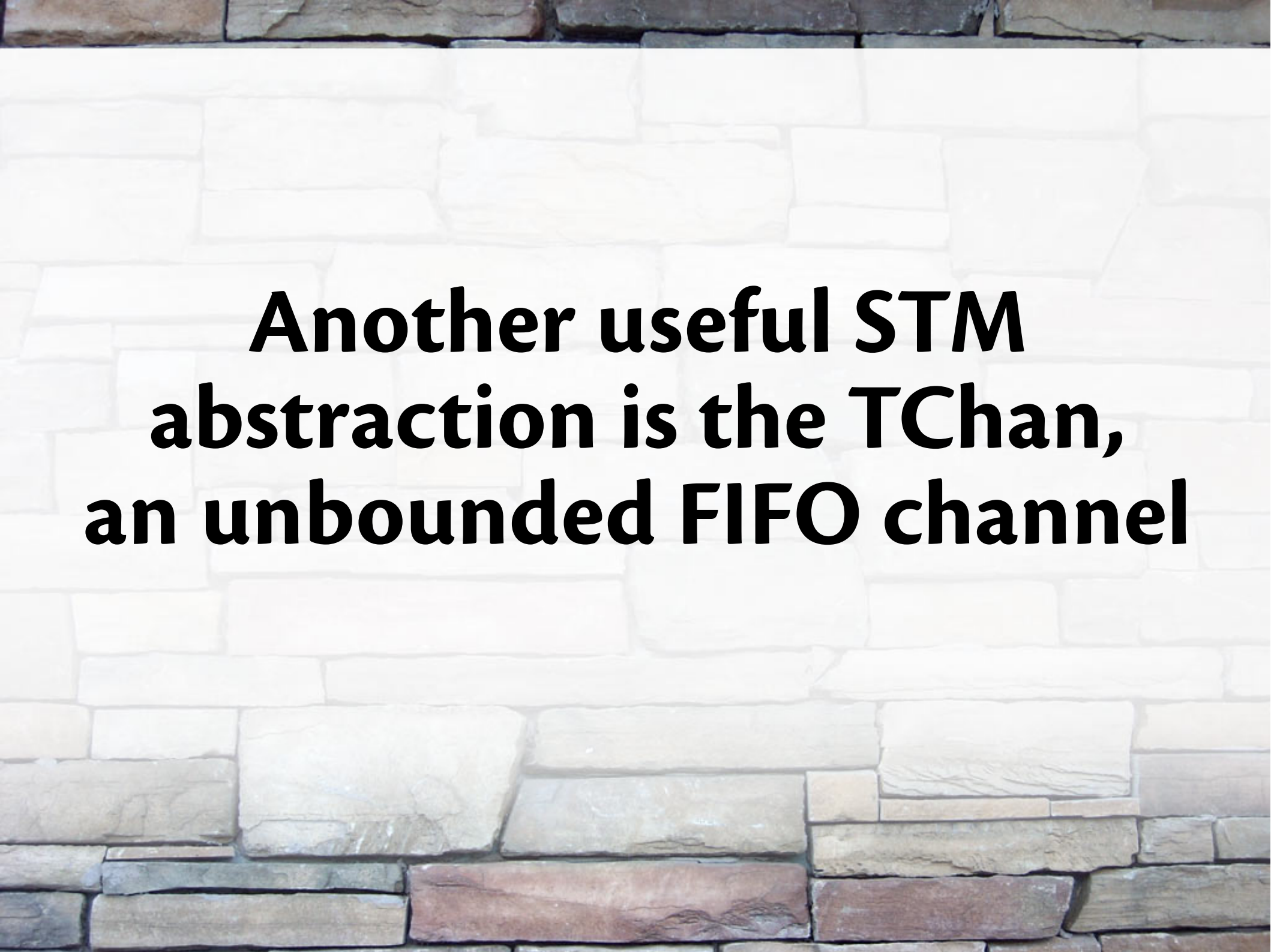
The marking is done through the type system at compile time

STM
is just another kind of
I/O
(with a different marker:
"STM a" instead of "IO a")

Transactional memory needs to be declared explicitly as TVar

The STM library provides an STM-to-IO converter called "atomically"

Transactional memory can only be accessed through dedicated functions like "modifyTVar", "readTVar", "writeTVar" which can only be called inside STM blocks


Another useful STM
abstraction is the TChan,
an unbounded FIFO channel

Once some messages are transferred into a TChan, they are ready to be consumed by other threads (broadcasting is possible too)

TChans are useful when threads need to send signals to each other, as opposed to just accessing shared state

Compile your STM code with:

```
ghc -threaded program.hs
```

When running the program:

```
./program +RTS -N
```


Follow me on GitHub
github.com/dserban