Local Linearizability

Ana Sokolova

joint work with:

Andreas Haas
Andreas Holzer
Michael Lippautz
Ali Sezgin

Tom Henzinger
Christoph Kirsch
Hannes Payer
Helmut Veith
Concurrent Data Structures
Correctness and Performance
Semantics of concurrent data structures

- **Sequential specification** = set of legal sequences

  - e.g. pools, queues, stacks

  - e.g. queue legal sequence: \texttt{enq(1)enq(2)deq(1)deq(2)}

- **Consistency condition** = e.g. linearizability / sequential consistency

  - e.g. the concurrent history above is a linearizable queue concurrent history
Consistency conditions

**Linearizability** [Herlihy, Wing '90]

- There exists a legal sequence that preserves precedence.

**Sequential Consistency** [Lamport '79]

- There exists a legal sequence that preserves per-thread precedence (program order).
Performance and scalability

throughput

# of threads / cores

:-))))

:-)

:-(

:-\
Relaxations allow trading correctness for performance.
Relaxing the Semantics

- **Sequential specification** = set of legal sequences
- **Consistency condition** = e.g. linearizability / sequential consistency

Quantitative relaxations
Henzinger, Kirsch, Payer, Sezgin, S. POPL13

Local linearizability
in this talk

for queues only
(feel free to ask for more)

not “sequentially correct”

too weak
Local Linearizability

main idea

• Partition a history into a set of local histories

• Require linearizability per local history

Already present in some shared-memory consistency conditions (not in our form of choice)

no global witness

Local sequential consistency… is also possible
Local Linearizability (queue) example

(t1-induced history, linearizable)
(t2-induced history, linearizable)

Locally linearizable

(sequential) history not linearizable
Local Linearizability (queue) definition

Queue signature $\Sigma = \{\text{enq}(x) \mid x \in V\} \cup \{\text{deq}(x) \mid x \in V\} \cup \{\text{deq}(\text{empty})\}$

For a history $h$ with a thread $T$, we put

$$I_T = \{\text{enq}(x)^T \in h \mid x \in V\}$$

$$O_T = \{\text{deq}(x)^T \in h \mid \text{enq}(x)^T \in I_T\} \cup \{\text{deq}(\text{empty})\}$$

$h$ is locally linearizable iff every thread-induced history $h_T = h \upharpoonright (I_T \cup O_T)$ is linearizable.
Local Linearizability for Container-Type DS

Signature $\Sigma = \text{Ins} \cup \text{Rem} \cup \text{SOb} \cup \text{DOb}$

For a history $h$ with a thread $T$, we put

$I_T = \{ m^T \in h \mid m \in \text{Ins} \}$

$O_T = \{ m(a) \in h \cap \text{Rem} \mid i(a)^T \in I_T \} \cup \{ m(e) \mid e \in \text{Emp} \}$

$\cup \{ m(a) \in h \cap \text{DOb} \mid i(a)^T \in I_T \}$

$h$ is locally linearizable iff every thread-induced history $h_T = h \mid (I_T \cup O_T)$ is linearizable.
Where do we stand?

In general

Local Linearizability

Linearizability

Sequential Consistency
Where do we stand?

For queues (and most container-type data structures)

Local Linearizability

Linearizability

Sequential Consistency
Local linearizability is compositional

\( h \) (over multiple objects) is locally linearizable iff each per-object subhistory of \( h \) is locally linearizable

Local linearizability is modular / “decompositional”

may allow for modular verification

like linearizability
unlike sequential consistency

uses decomposition into smaller histories, by definition
Verification (queue)

Queue sequential specification (axiomatic)

$s$ is a legal queue sequence
iff
1. $s$ is a legal pool sequence, and
2. $\text{enq}(x) <_s \text{enq}(y) \land \text{deq}(y) \in s \implies \text{deq}(x) \in s \land \text{deq}(x) <_s \text{deq}(y)$

Queue linearizability (axiomatic)

$h$ is queue linearizable
iff
1. $h$ is pool linearizable, and
2. $\text{enq}(x) <_h \text{enq}(y) \land \text{deq}(y) \in h \implies \text{deq}(x) \in h \land \text{deq}(y) <_h \text{deq}(x)$

precedence order

Henzinger, Sezgin, Vafeiadis CONCUR13
Verification (queue)

Queue sequential specification (axiomatic)

$s$ is a legal queue sequence
iff
1. $s$ is a legal pool sequence, and
2. $\text{enq}(x) <_s \text{enq}(y) \land \text{deq}(y) \in s \Rightarrow \text{deq}(x) \in s \land \text{deq}(x) <_s \text{deq}(y)

Queue local linearizability (axiomatic)

$h$ is queue locally linearizable
iff
1. $h$ is pool locally linearizable, and
2. $\text{enq}(x) <_h \text{enq}(y) \land \text{deq}(y) \in h \Rightarrow \text{deq}(x) \in h \land \text{deq}(y) <_h \text{deq}(x)$

thread-local precedence order
Generic Implementations

Your favorite linearizable data structure implementation

turns into a locally linearizable implementation by:

- LLD $\Phi$ (locally linearizable)
- LL+D $\Phi$ (also pool linearizable)

segment of possibly dynamic size (n)

local inserts / global (randomly distributed) removes
Performance

![Graph showing performance metrics for different data structures and thread counts.]

(a) Queues, LL queues, and “queue-like” pools

LL+D MS queue performs significantly better than MS queue.
Performance

Figure 8: Performance and scalability of producer-consumer microbenchmarks with an increasing number of threads on a 40-core (2 hyperthreads per core) machine. LLD \( \Phi \) performs significantly better than \( \Phi \).

(a) Queues, LL queues, and “queue-like” pools

Local linearizability utilizes the idea of decomposing a history into a set of thread-induced histories and requiring consistency of observations in history. Then, given a set of insert operations that happen or not. ExLL enables the introduction of fine-grained synchronization between (subsets of) threads. An insert-operation is justified if and only if it is appertinent to the transactional process.

We believe that ExLL is a very promising research direction as it enables new programming styles. Moreover, it shows the virtue of the simplicity of local linearizability’s definition. A theoretical investigation of ExLL and efficient implementations of corresponding instantiation of an observer method. For ease of presentation, we have considered a global parameter \( \Phi \). How-
(a) Queues, LL queues, and “queue-like” pools
Local Linearizability

Ana Sokolova

joint work with:

Andreas Haas Google
Andreas Holzer University of Toronto
Michael Lippautz Google
Ali Sezgin University of Cambridge

Tom Henzinger IST Austria
Christoph Kirsch University of Salzburg
Hannes Payer Google
Helmut Veith TU Wien

Thank You!
Local Linearizability

Ana Sokolova

joint work with:

Andreas Haas  Google
Andreas Holzer
Michael Lippautz  Google
Ali Sezgin

Tom Henzinger
Christoph Kirsch
Hannes Payer  Google
Helmut Veith  TU Wien