Memory Consistency for Parallel Systems: A Reformulation Without Global Time

Jonathan Z.Y. Hay Y.C. Tay
Department of Computer Science and Department of Mathematics
National University of Singapore

Abstract— **Cross-chip latencies now make multi-core architectures resemble distributed systems. The design of distributed protocols is notoriously error-prone, particularly when their analysis is based on the use of global time. Classical memory consistency models for parallel programming, such as linearizability, uses such a global ordering. This talk examines the reformulation, without global time, of these consistency models.**

I. Introduction

It is now common for a processor chip to have multiple cores and caches. Furthermore, current processor speeds are so fast that it takes many cycles for a signal to travel across a chip. These make a chip increasingly resemble a distributed system.

The design and analysis of distributed protocols is notoriously prone to error. In trying to understand why this is so [4], we learnt two lessons.

The first is that many errors originate from our habit to reason (often subconsciously) using global time, or some global interleaving order of all events in the system, i.e. to think sequentially about a parallel execution.

The theory (definitions and proofs) for distributed protocols should instead rely only on partial orderings of the events. This applies to the theory for parallel processing as well, now that they behave like a distributed system.

In the case of consistency models for shared memory, the classical theory starts with a total order of all events in the system. Two well-known models are sequential consistency and linearizability, and the need for global time distinguishes these two definitions.

II. Sequential Consistency

For notational simplicity, we assume every process (or thread) executes a totally ordered

sequence of operations, and the **process order** $\prec^{\mathrm{o}}_{\mathrm{P}}$ is the union of these total orders. Following Steinke and Nutt [3], the only operations are reads and writes, and each write generates a unique value.

Let $w^x_B(v)$ denote the operation where process B writes value v to variable x; similarly, $r^x_B(v)$ denotes B reading value v of x. We say $r^x_C(v)$ **reads from** $w^x_B(v)$, denoted $w^x_B(v) \mapsto r^x_C(v)$ if and only if the value read by C was written by B. We call $\prec^{\mathrm{o}}_{\mathrm{P}} \cup \mapsto$ an **operation history**.

A total order $<^{\mathrm{o}}$ on the operations is **legal** if and only if whenever $w^x_B(v) <^{\mathrm{o}} r^x_C(v)$, there is no $w^x_A(u)$ such that $w^x_B(v) <^{\mathrm{o}} w^x_A(u) <^{\mathrm{o}} r^x_C(v)$.

For a partial order $\prec^{\mathrm{o}}$ on operations, **SerialView**$(\prec^{\mathrm{o}})$ denotes a legal total order $<^{\mathrm{o}}$ that preserves $\prec^{\mathrm{o}}$, i.e. $\prec^{\mathrm{o}} \subseteq <^{\mathrm{o}}$.

An operation history is **sequentially consistent** if and only if $\exists$SerialView$(\prec^{\mathrm{o}}_{\mathrm{P}})$.

Steinke and Nutt used such a formalism to express several other correctness criteria — PRAM consistent, processor consistent, causally consistent, etc. — all without using global time. Can linearizability be similarly defined?

Linearizability is fundamentally different from sequential consistency in that it is a *local* property, i.e. the operation history is linearizable if and only if it is linearizable for every object. Sequential consistency is a weaker criterion that does not have such a property.

Note that $\exists$SerialView$(\prec^{\mathrm{o}}_{\mathrm{P}})$ does not include the reads from order $\mapsto$ defined on objects. We can further define a **data order** $\prec^x_{\mathrm{D}}$ for each variable x, as follows: If $o^x_C(u) \prec^{\mathrm{o}}_{\mathrm{P}} r^x_C(v)$, $r^x_C(v)$ reads from $w^x_B(v)$ and $u \neq v$, then $o^x_C(u) \prec^x_{\mathrm{D}} w^x_B(v)$.

Let $\prec^{\mathrm{o}}_{\mathrm{D}}$ be the transitive closure of $\prec^{\mathrm{o}}_{\mathrm{P}} \cup \mapsto \cup (\bigcup_x \prec^x_{\mathrm{D}})$. An operation history is **data consistent** if and only if $\exists$SerialView$(\prec^{\mathrm{o}}_{\mathrm{D}})$.

Theorem

An operation history is data consistent if and only if it is sequentially consistent.

In other words, adding $\mapsto$ and $\prec^x_{\mathrm{D}}$ to $\prec^{\mathrm{o}}_{\mathrm{P}}$ does not give a correctness criterion that is stronger than sequential consistency.

III. Linearizability

Since we assume a process B executes sequentially, we can totally order events at B with some local B-time. An operation $o^x_B(v)$ thus spans a B-time interval between two events: its **invocation** $Inv(o^x_B(v))$ and the **response** $Resp(o^x_B(v))$. Current cross-chip latencies make such intervals nontrivial, so operations are not atomic.

Classically, linearizability is defined by starting with a global total order (using "real time" [2]) of all events and extracting a partial order on operations from that total ordering on

events. Can linearizability be defined without such a total ordering?

Define a **causal order** $\prec_{\mathrm{c}}^{\mathrm{e}}$ on events thus: For two events f_B and f'_C at processes B and C, $f_B \prec_{\mathrm{c}}^{\mathrm{e}} f'_C$ if and only if f_B can causally affect f'_C.

For $B = C$, this means f_B happens before f'_B in B-time; for $B \neq C$, this means a signal sent at B-time for f_B can travel across the chip and reach C at some C-time before f'_C.

We call this causal order $\prec_{\mathrm{c}}^{\mathrm{e}}$ an **event history**.

An event history $\prec_{\mathrm{c}}^{\mathrm{e}}$ induces an **operation history** $\prec^{\mathrm{o}}$ where $o_B^x(u) \prec^{\mathrm{o}} o_C^y(v)$ if and only if $Resp(o_B^x(u)) \prec_{\mathrm{c}}^{\mathrm{e}} Inv(o_C^y(v))$.

For an event history $\prec_{\mathrm{c}}^{\mathrm{e}}$, the **process subhistory** $<_B^{\mathrm{e}}$ is the restriction of that order to events for process B; this restriction yields a total order since we assume a process is a sequence of operations.

Similarly, the **object subhistory** $\prec_x^{\mathrm{e}}$ is the restriction of that order to events for object x.

The standard definition of linearizability uses a total order $<_{\mathrm{g}}^{\mathrm{e}}$ (instead of $\prec_{\mathrm{c}}^{\mathrm{e}}$) imposed by global time. Two total orders $<_{\mathrm{g}}^{\mathrm{e}}$ and $<_{\mathrm{g}}^{\mathrm{e}\prime}$ are **equivalent**, denoted $<_{\mathrm{g}}^{\mathrm{e}} \equiv <_{\mathrm{g}}^{\mathrm{e}\prime}$, if and only if $<_B^{\mathrm{e}} = <_B^{\mathrm{e}\prime}$ for every process B.

$<_{\mathrm{g}}^{\mathrm{e}}$ is **sequential** if and only if the operation history that it induces is a total order. A sequential $<_{\mathrm{g}}^{\mathrm{e}}$ is **legal** if and only if the operation history that it induces is legal.

Classically, $<_{\mathrm{g}}^{\mathrm{e}}$ is **linearizable** if and only if there is some legal sequential $<_{\mathrm{g}}^{\mathrm{e}\prime}$ such that $<_{\mathrm{g}}^{\mathrm{e}} \equiv <_{\mathrm{g}}^{\mathrm{e}\prime}$ and $\prec^{\mathrm{o}} \subseteq \prec^{\mathrm{o}\prime}$, where $\prec^{\mathrm{o}}$ and $\prec^{\mathrm{o}\prime}$ are the operation histories induced by $<_{\mathrm{g}}^{\mathrm{e}}$ and $<_{\mathrm{g}}^{\mathrm{e}\prime}$ respectively.

One can prove that $<_{\mathrm{g}}^{\mathrm{e}}$ is linearizable if and only if $\prec_x^{\mathrm{e}}$ is linearizable for every object x; this is the local property mentioned in Section 2.

Golab has proposed two definitions of linearizability that do not use real time [1]. Using our notation, his definitions can be stated as:

(1) $\prec_{\mathrm{c}}^{\mathrm{e}}$ is **$\exists$-linearizable** if and only if there is a total order $<_{\mathrm{g}}^{\mathrm{e}}$ such that $\prec_{\mathrm{c}}^{\mathrm{e}} \subseteq <_{\mathrm{g}}^{\mathrm{e}}$ and $<_{\mathrm{g}}^{\mathrm{e}}$ is linearizable.

(2) $\prec_{\mathrm{c}}^{\mathrm{e}}$ is **$\forall$-linearizable** if and only if for every total order $<_{\mathrm{g}}^{\mathrm{e}}$ such that $\prec_{\mathrm{c}}^{\mathrm{e}} \subseteq <_{\mathrm{g}}^{\mathrm{e}}$, $<_{\mathrm{g}}^{\mathrm{e}}$ is linearizable.

Golab conjectured that the first definition is not a local property, but the second definition is.

We have found counterexamples to show that $\exists$-linearizability is indeed not local, so it is arguably not the right generalization of linearizability. We have also proven Golab's conjecture for $\forall$-linearizability:

Theorem

$\prec_{\mathrm{c}}^{\mathrm{e}}$ is $\forall$-linearizable if and only if $\prec_x^{\mathrm{e}}$ is $\forall$-linearizable for every object x.

IV. Conclusion

Although $\forall$-linearizability is a local property, we think it is also not the right generalization. Our skepticism is based on the second lesson that we learnt from distributed computing, namely: Processes and objects are asymmetric in their properties, so it makes a difference whether a definition is in terms of events at processes or at objects.

The classical definition for linearizability is in terms of events that model the non-atomicity of operations, so the events are all local to processes. However, there are actually four events associated with each $o_B^x(v)$: $Inv(o_B^x(v))$ at B, the event at x for receiving the invocation, the event at x for sending the response, and $Resp(o_B^x(v))$ at B. Like the interval between $Inv(o_B^x(v))$ and $Resp(o_B^x(v))$, the delay between the receive and send events at x may also be nontrivial (consider, say, a cache miss).

A proper reformulation of consistency for shared memory should therefore model events at both processes and objects, and relate them through a partial order defined with local times for all events.

Acknowledgment

We thank Wojciech Golab and Seth Gilbert for their helpful comments.

References

[1] W. Golab. Relativistic linearizability. (Private communication through Seth Gilbert.), Feb. 2012.

[2] C. Shao, J. L. Welch, E. Pierce, and H. Lee. Multiwriter consistency conditions for shared memory registers. *SIAM J. Comput.*, 40(1):28–62, 2011.

[3] R. C. Steinke and G. J. Nutt. A unified theory of shared memory consistency. *J. ACM*, 51(5):800–849, Sept. 2004.

[4] Y. C. Tay and W. T. Loke. On deadlocks of exclusive AND-requests for resources. *Distrib. Comput.*, 9(2):77–94, Oct. 1995.