Errata Slip

Proceedings of the 2017 USENIX Annual Technical Conference

In the paper "StreamBox: Modern Stream Processing on a Multicore Machine" by Hongyu Miao and Heejin Park, *Purdue ECE;* Myeongjae Jeon and Gennady Pekhimenko, *Microsoft Research;* Kathryn S. McKinley, *Google;* Felix Xiaozhu Lin, *Purdue ECE* (Thursday session, "Multicore," pp. 617–629 of the Proceedings) the text at the top of page 621 in the left-hand column does not display correctly.

Original text

WM_CONSUMED After D_{own} consumes the end wat mark, it guarantees that it has flushed all derived st and the end watermark to the downstream contai and D_{own} may be destroyed.

WM_CANCELLED D_{up} chooses not to emit the end v termark for the (potential) epoch. Section 5.2 scribes how we support windowing transforms by c celling watermarks and merging containers.

Lock-free container processing Containers are lo free to minimize synchronization overhead. We insta ate the end watermark as an atomic variable that enfor acquire-release memory order. It ensures that D_{own} serves all D_{up} evaluators' writes to the container's claimed bundle set before observing D_{up}'s write of end watermark. The unclaimed bundle set is a concurr data structure that aggressively weakens the ordering mantics on bundles for scalability. Examples of ot such data structures include non-linearizable lock-f queues [13] and relaxed priority queues [4]. We furtner exploit this flexibility to make the bundle set NUMA-aware, as discussed in Section 7.1.

Corrected text

WM_CONSUMED After D_{own} consumes the end watermark, it guarantees that it has flushed all derived state and the end watermark to the downstream container and D_{own} may be destroyed.

WM_CANCELLED D_{up} chooses not to emit the end watermark for the (potential) epoch. Section 5.2 describes how we support windowing transforms by cancelling watermarks and merging containers.

Lock-free container processing Containers are lock-free to minimize synchronization overhead. We instantiate the end watermark as an atomic variable that enforces acquire-release memory order. It ensures that D_{own} observes all D_{up} evaluators' writes to the container's unclaimed bundle set before observing D_{up}'s write of the end watermark. The unclaimed bundle set is a concurrent data structure that aggressively weakens the ordering semantics on bundles for scalability. Examples of other such data structures include non-linearizable lock-free queues [13] and relaxed priority queues [4]. We further exploit this flexibility to make the bundle set NUMA-aware, as discussed in Section 7.1.