Tiny Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs

Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A. Chien, and Haryadi S. Gunawi

THE UNIVERSITY OF CHICAGO

UCARE ucare.cs.uchicago.edu

CERES Center for Unstoppable Computing ceres.cs.uchicago.edu

GC-Induced Tail Latencies

Google: Taming The Long Latency Tail - When More Machines Equals Worse Results

*"[If a] read is stuck behind an erase, [it] must wait 10s of ms, … a **100x** increase in latency variance"*

Why SSDs don't perform

From their earliest days, people have reported that SSDs were not providing the performance they expected. As SSDs age, for instance, they get slower. Here's why.

Why it's hard to meet SLAs with SSDs

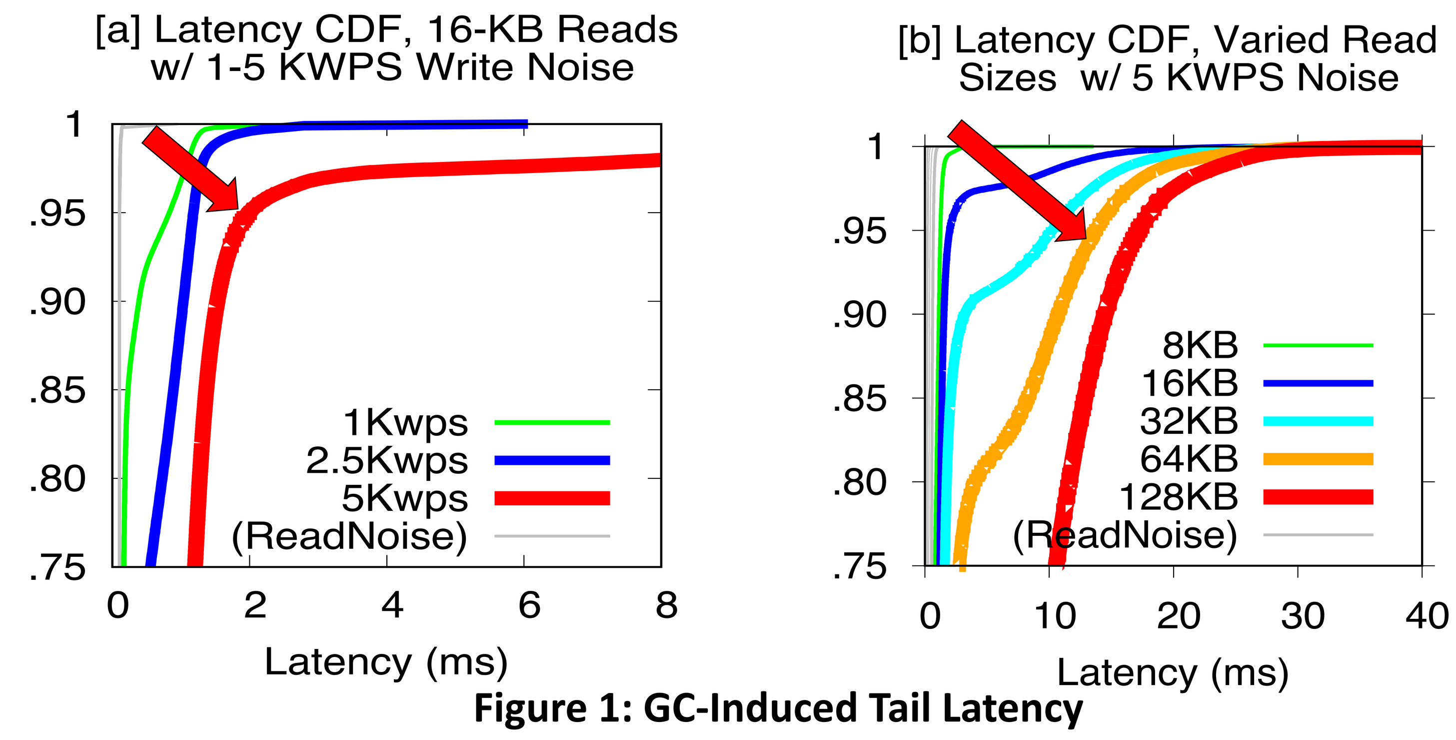

Figure 1: GC-Induced Tail Latency

More frequent GCs block incoming reads (from more intense random writes) and create longer tail latencies.

As read size increases, the probability of one of the pages being blocked by GC also increases.

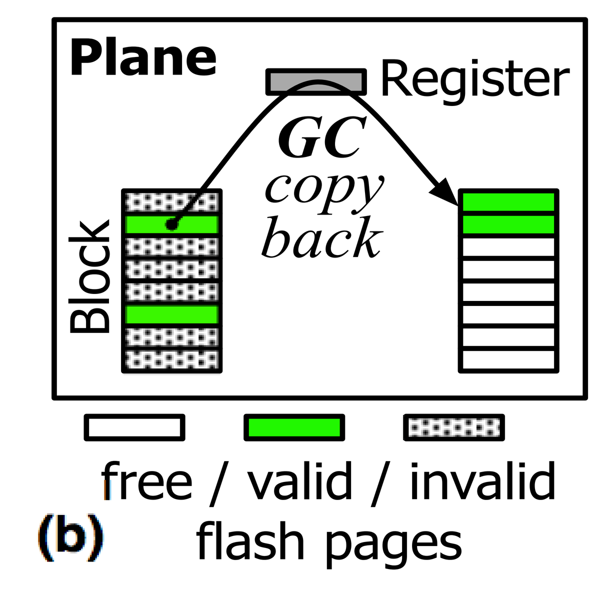

Figure 2: Example of GC Copyback

GC has to copy forward all valid pages to new block before erasing old block.

Tiny Tail Flash (ttFlash) Architecture

Leverage three major SSD technological advancements:

- Increasing power and speed of today's flash controller
- Redundant Array of Independent NAND (RAIN)
- "Super capacitor" backed RAM

1. Plane-Blocking GC(**PB**): block GCing-plane (finer granularity)
2. GC-Tolerant Read(**GTR**): XOR to reconstruct page blocked by GC
3. GC-Tolerant Flush(**GTF**): store write blocked by GC in RAM
4. Rotating GC(**RGC**): limit GCing-plane to 1 per RAID group

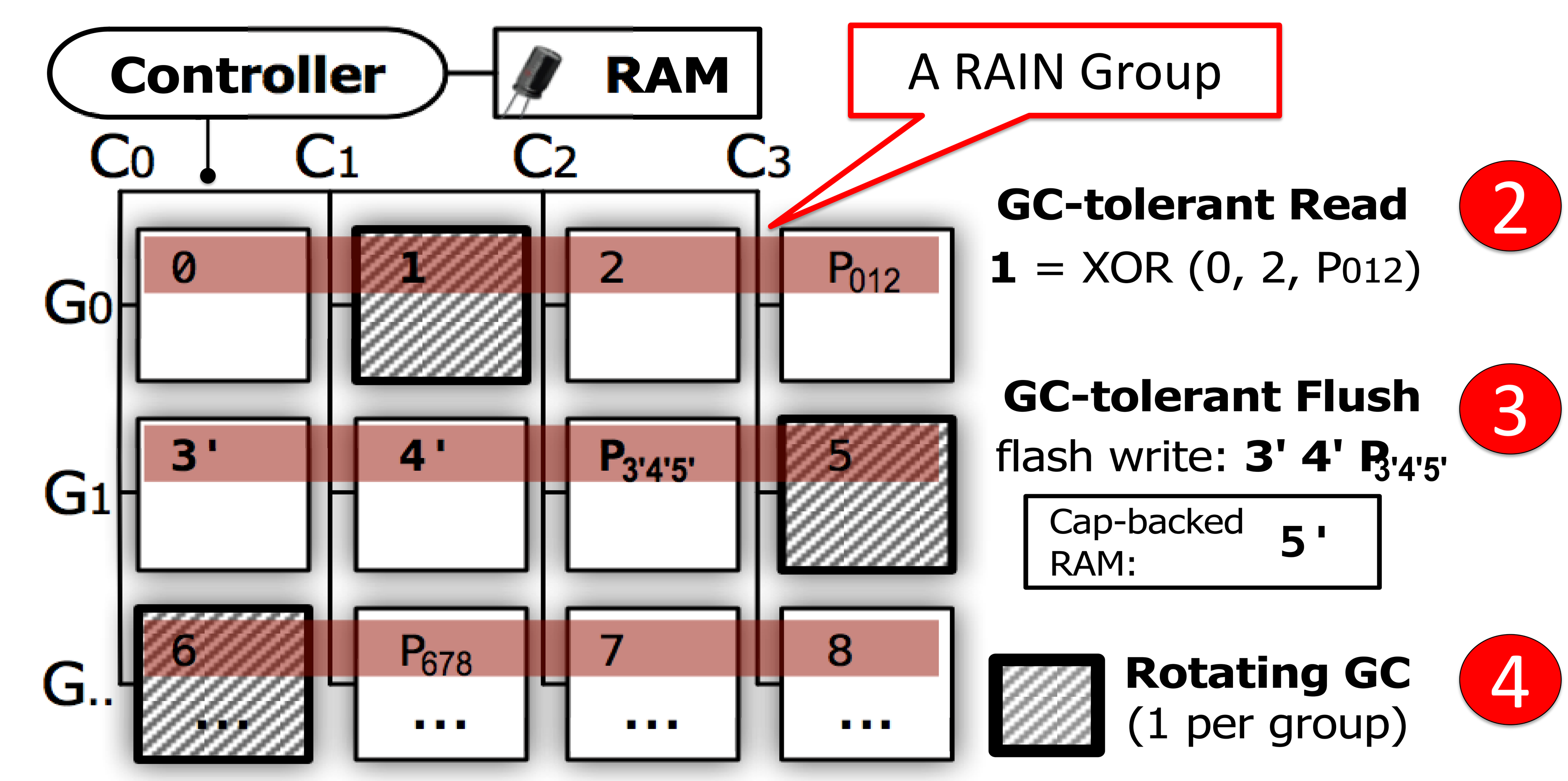

Figure 3: ttFlash Architecture

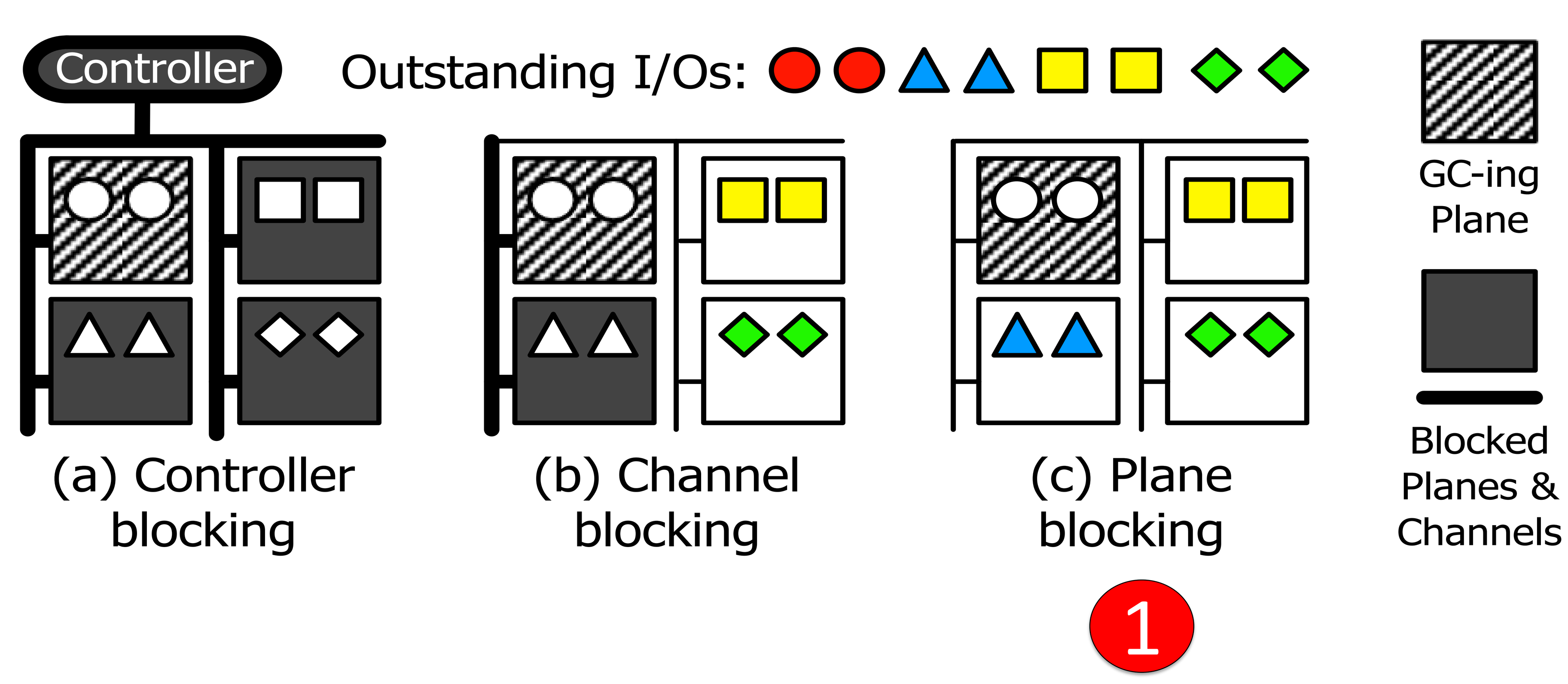

Figure 4: Various Levels of GC-Blocking

Experiment Results

Evaluation with 6 real-world traces (Windows servers)

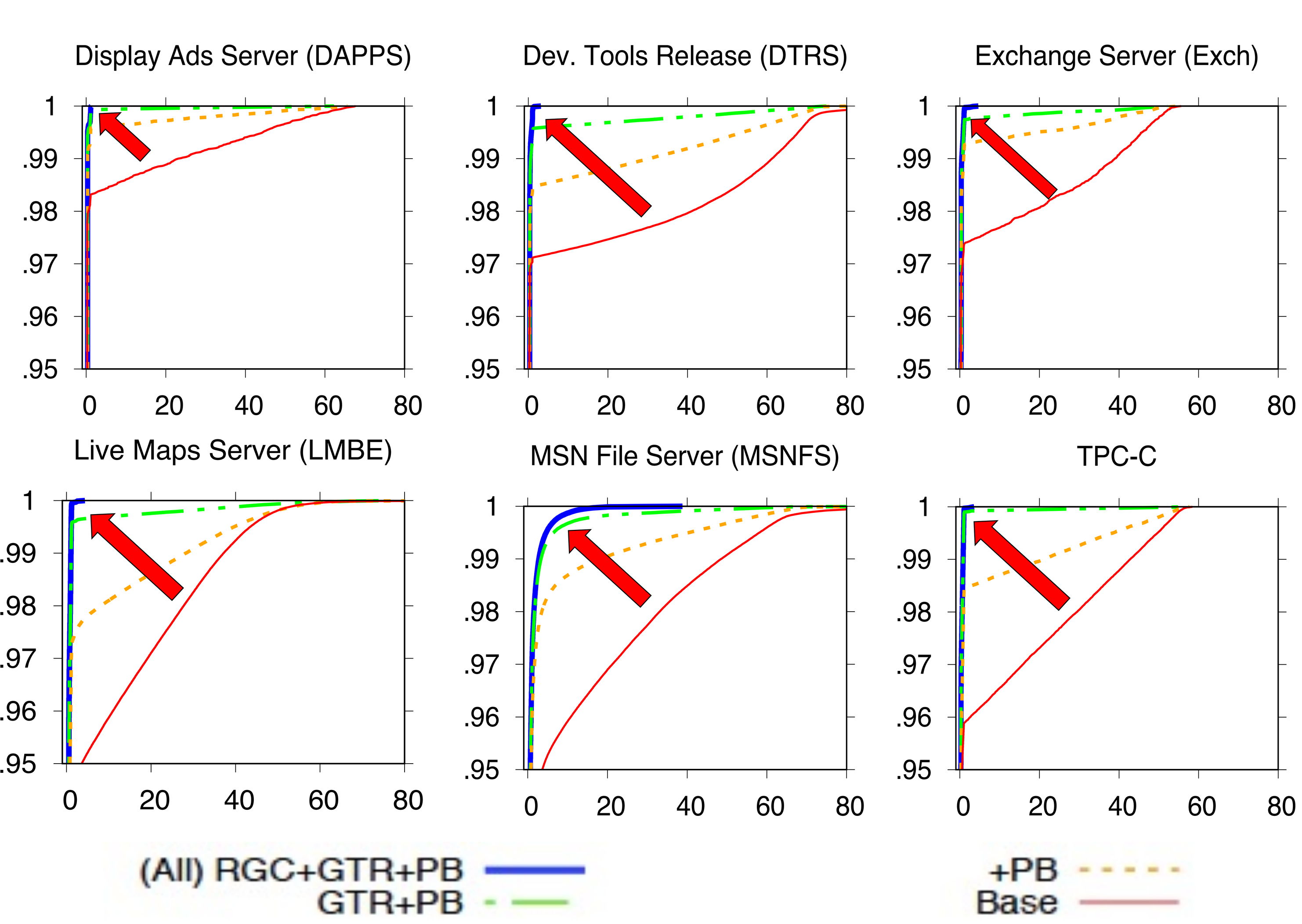

Figure 5: CDF of Read Latencies

Percentile	DAP	DTRS	EXCH	LMBE	MSN	TPCC
99.99th	1.0x	1.2	1.2	2.0	1.0	2.6
99.9th	1.0x	1.0	1.0	1.0	1.0	1.0
99th	1.0x	1.0	1.1	1.0	1.0	1.0

Table 1: ttFlash vs. NoGC

99 – 99.9th: < ***1.1x*** for ttFlash and < ***138.2x*** for Base
99.99th: < ***2.6x*** for ttFlash and < ***91.9x*** for Base

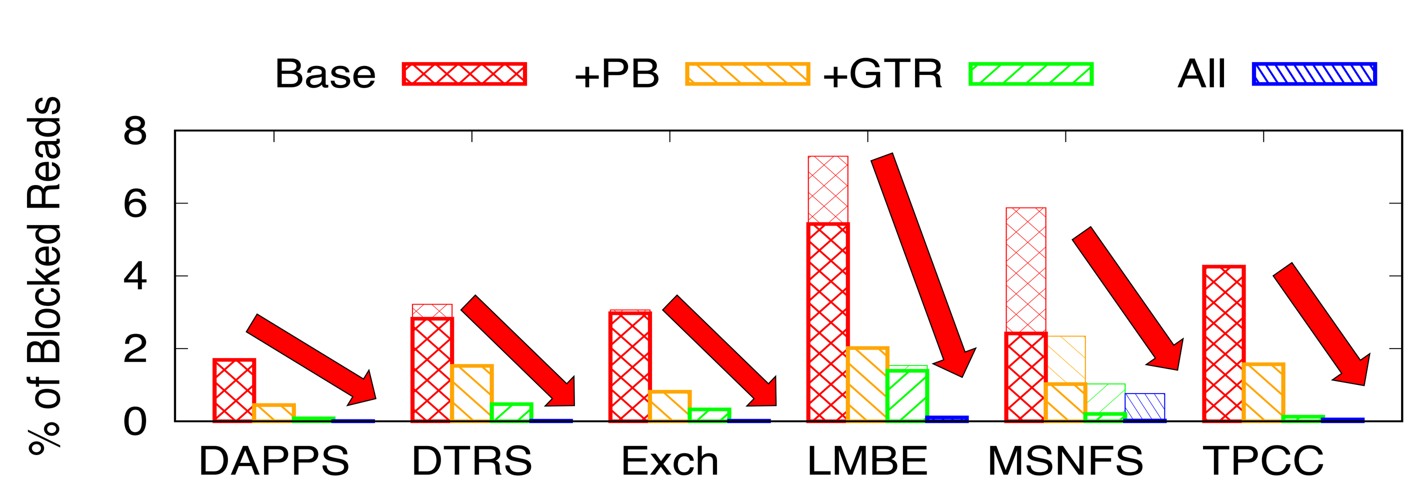

Figure 6: % GC-blocked Read I/Os

Reduced blocked I/Os (total) from ***2 – 7%*** to ***0.003 – 0.7%***

Average Latencies:

ttFlash is *2.52-7.88x* faster than Base (with RAIN) and *1.09-1.33x* slower than NoGC (without RAIN).

GC Overheads:

ttFlash introduces *15 – 18%* of additional P/E cycles (in 4 out of 6 workloads) due to RAIN (Ideally *1/7 ≈15%*)

Appeared in the 15th USENIX Conference on File and Storage Technologies (FAST), 2017