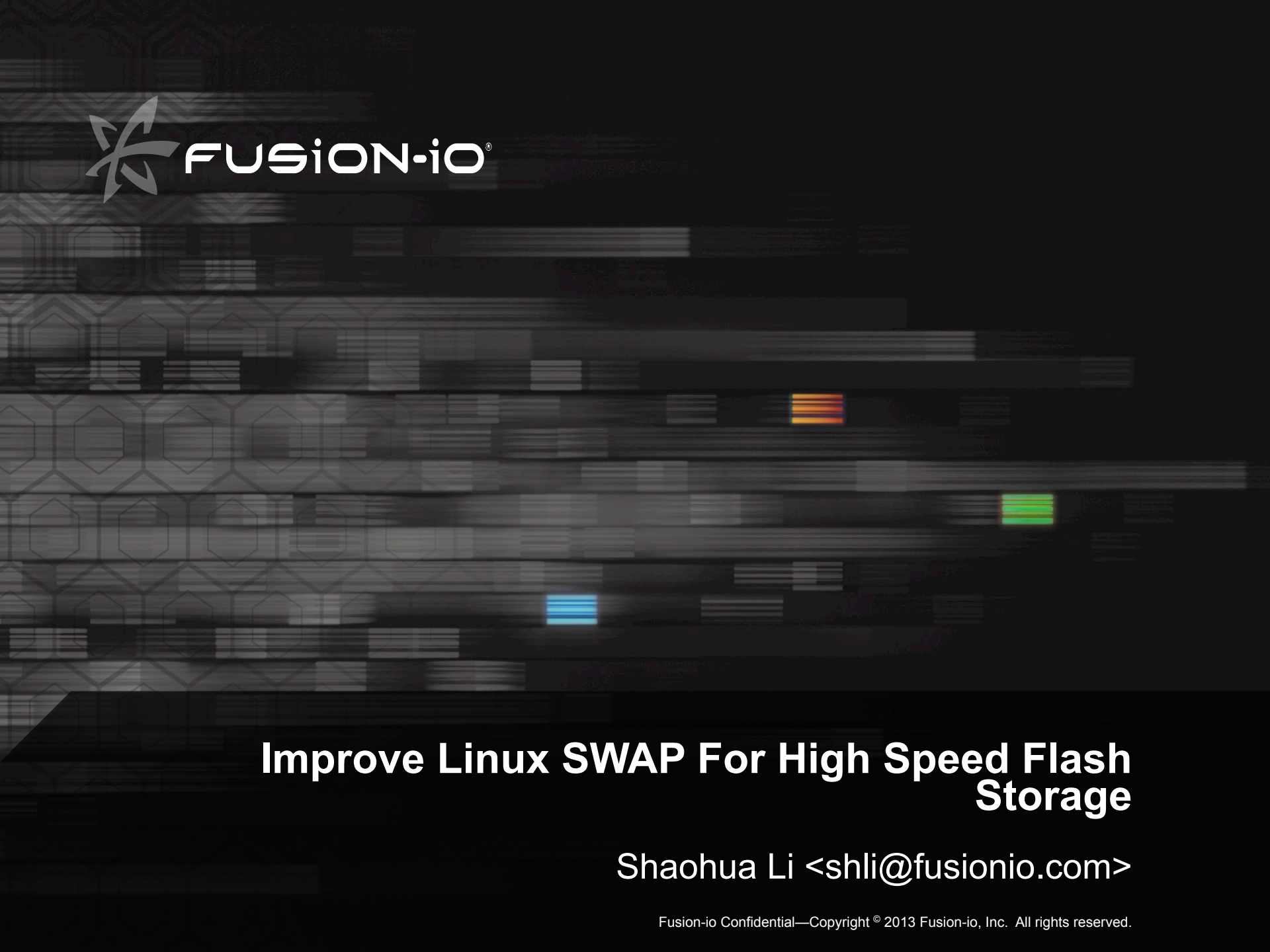


Improve Linux SWAP For High Speed Flash Storage

Shaohua Li <shli@fusionio.com>

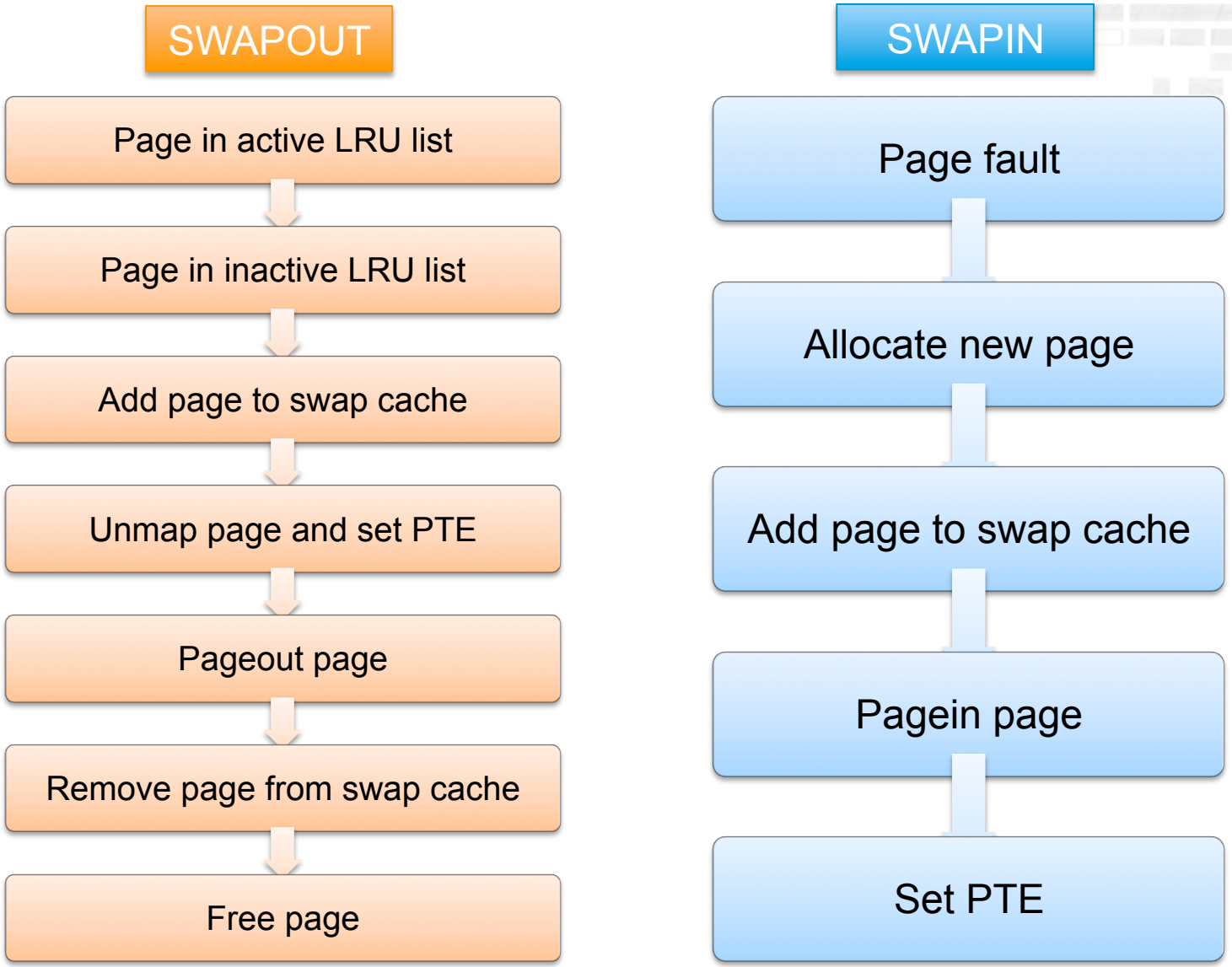
Background

FUSION-IO

- Partially replace DRAM with Flash storage (SSD)
- Compared to DRAM, Flash has:
  - Low price
  - High density
  - Low power
- Reasonable latency is tolerable

How does swap work?

FUSION-IO



SSD specific characteristics

FUSION-IO

- Fast
- No seek penalty
- Big size request has better throughput
- High iodepth has better throughput
- Discard

SWAPOUT – TLB flush

FUSION-IO

- At least 2 TLB flushes
  - Clear PTE ‘A’ bit
  - Clear PTE ‘P’ bit
- Overhead is quite high
  - TLB flush is page based
  - TLB flush is involved by several tasks
- Solution:
  - Improve smp_call_function_many()
  - The first TLB flush can be removed in x86?
  - Batch TLB flush?

SWAPOUT – swap_map scan

FUSION-IO

- swap_map entry - in-memory data structure to track swap disk usage
- Slow linear memory scan to find a cluster (128 adjacent pages) - to produce big size IO request
- Solution: cluster list
  - Pros - O(1) algorithm
  - Cons - restrictive cluster alignment
  - Only enabled for SSD

SWAPOUT – IO pattern

FUSION-IO

- Interleaved IO pattern
  - Multiple reclaimers
  - New found cluster is shared by all reclaimers
- Block layer can't merge the interleaved IO completely
- Solution: per-cpu cluster
  - Reclaimer does sequential IO
  - Easy to do IO merge in block layer

SWAPIN

FUSION-IO

- Page fault does sync IO - iodepth 1, page size IO request
- Need swap readahead to produce:
  - Big size IO request
  - High iodepth IO
  - Parallel IO and CPU
- Userspace readahead API
  - madvise(MADV_WILLNEED) is extended to do swap prefetch

SWAPIN - cont

FUSION-IO

- In-kernel readahead
  - Arbitrary readahead (always 8 pages)
  - Random access workload
    - Unnecessary currently
    - Waste IO and increase memory pressure
    - Let readahead aware workload is random
  - Sequential access workload
    - Not enough currently
    - Hard to do - can’t guarantee sequentially accessed pages swapped out sequentially
      - Sequentially accessed pages might not live adjacently in LRU list
      - Adjacent pages of LRU list might be swapout by different reclaimers

Lock contentions

FUSION-IO

- Lock contentions are high
  - Concurrent swapout - kswapd, direct page reclaim
  - Concurrent swapin – page fault from each task
- Solution:
  - anon_vma mutex - now it's a rw_semaphore
  - swap_lock and swap address space lock
    - Per-swap lock now (a workaround)
    - Still have lock contention with very high speed SSD

SWAP discard

FUSION-IO

- Discard is important to optimize SSD write throughput
- swap discard implementation is synchronous
  - Block layer discard API is sync (introduce delay)
  - Discard just before write is useless
- Solution: async swap discard
  - No delay
  - Discard and write can run in the same time
  - Discard is cluster based

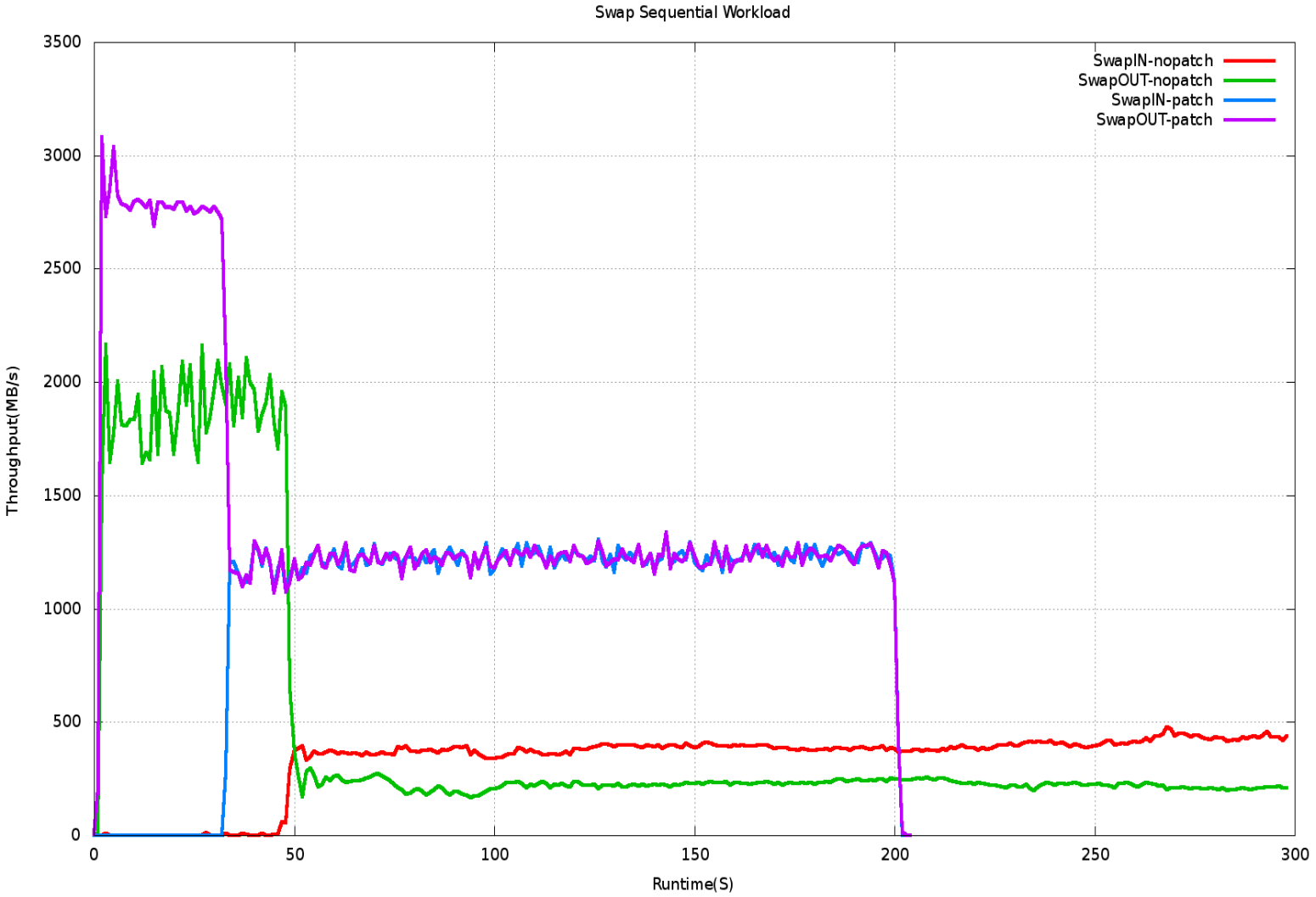
Other issues

FUSION-IO

- Page reclaim policy – bias swap?
- Huge page swap

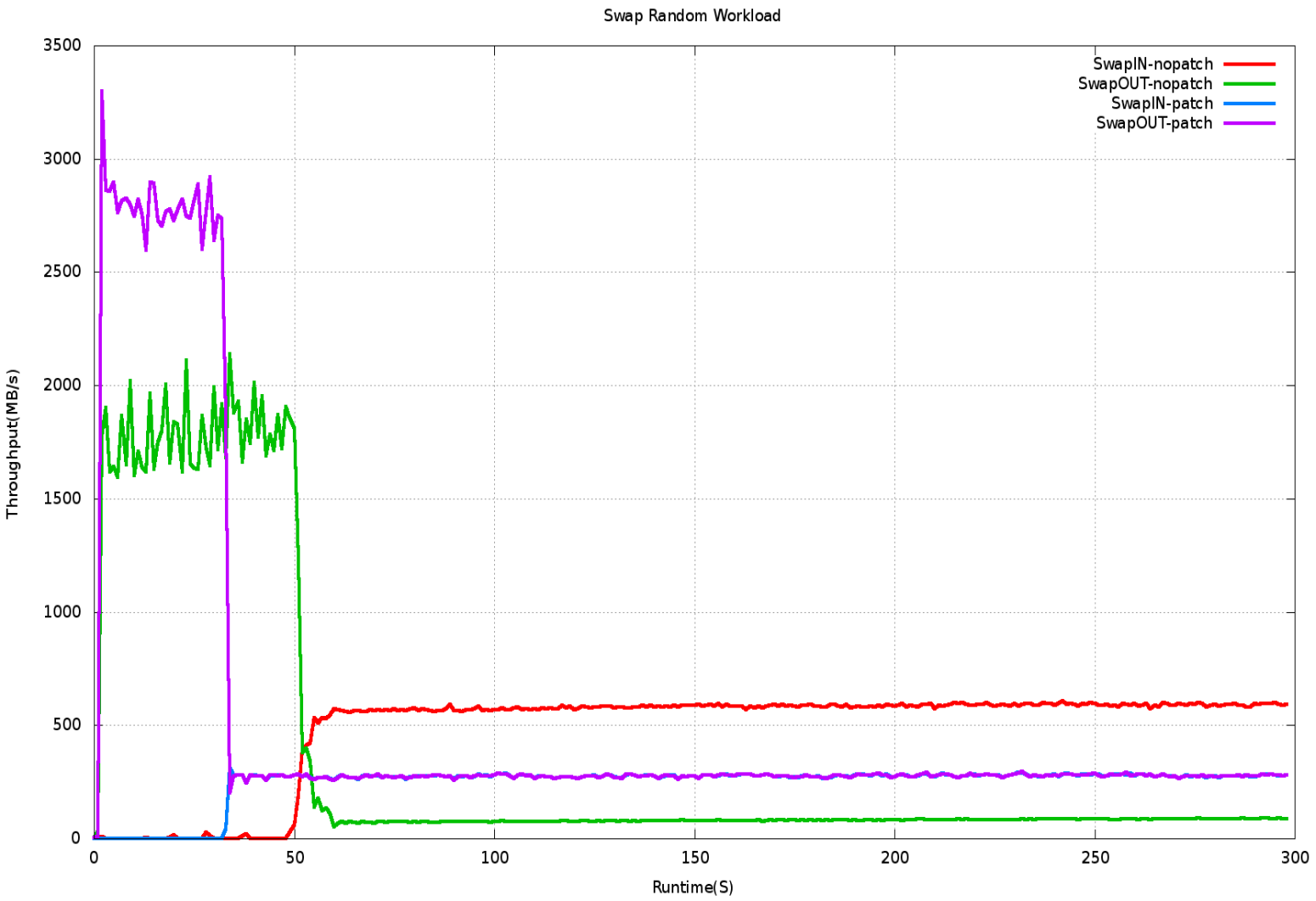
Benchmark – sequential workload

FUSION-IO



Benchmark – random workload

FUSION-IO




fusionio.com | REDEFINE WHAT'S POSSIBLE