FSWD
SAMSUNG

Performance Modeling of SSD

Seongjun Ahn, Dongjun Shin

Samsung Electronics

Content

- Background
- Modeling – Basic & Advanced
- Performance Metric
- Performance Estimation
- Future Work
- Conclusion

Introduction

- **Design parameters of SSD**
  - **SSD architecture**
    - Computing: CPU clock, etc
    - I/O: number of channels & banks
  - **NAND flash memory**
    - tRE/tWE, tR, tProg, tBER
- **Why do performance modeling?**
  - **To estimate performance of changing architecture and NAND**
  - **To understand the impact of changes of design parameters**
- **Performance metrics**
  - **Sequential I/O bandwidth (MB/s)**
  - **Random IOPS**

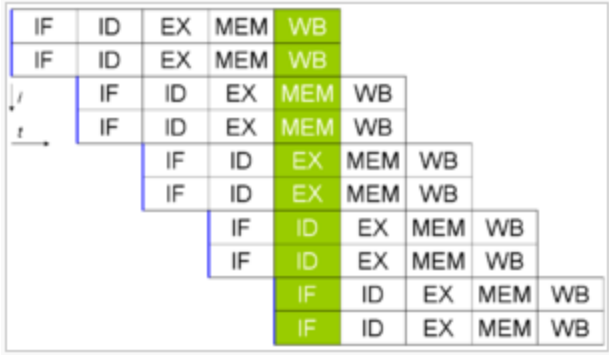
Previous Works - ILP

- **ILP (Instruction-Level Parallelism)**
  - **RISC instruction pipelining**
  - **Throughput = 1/L instruction/sec**
    - L is the latency of a stage (usually clock cycle)

- **How to increase throughput?**
  - **Deeper pipeline → smaller L**
  - **Superscalar pipelining → throughput is N/L**

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

<Instruction pipelining>



<Superscalar pipelining>

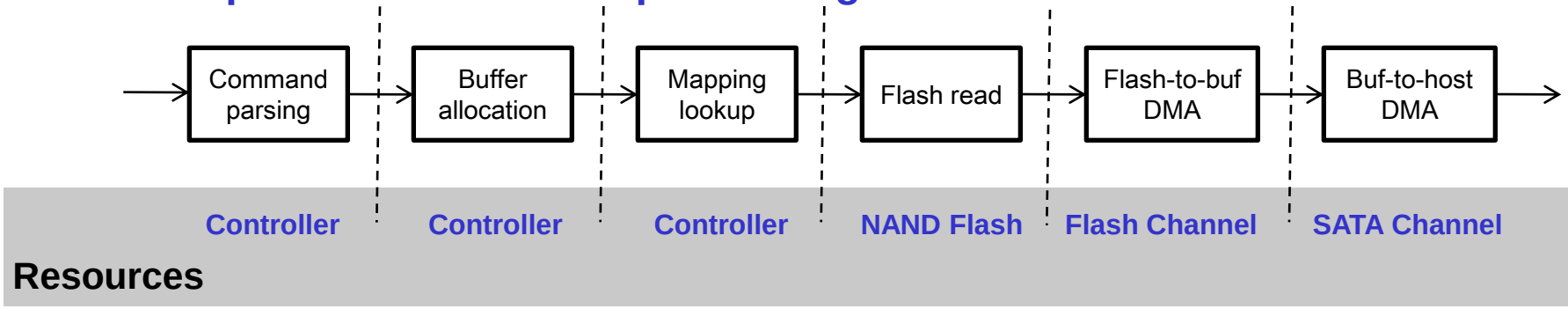
* All images in this slide are from wikipedia

ILP vs. SSD

- **SSD operation is also pipelined**
  - **Parallelism of computation and I/O**
  - **Resources and latencies of each stage are different**

- **How to model pipeline with asymmetric configuration?**

Example. Read command processing



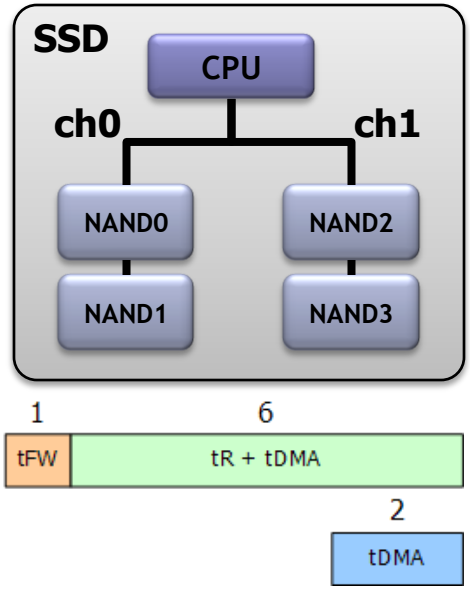
<from presentation of D.G.Lee, NVRAMOS08>

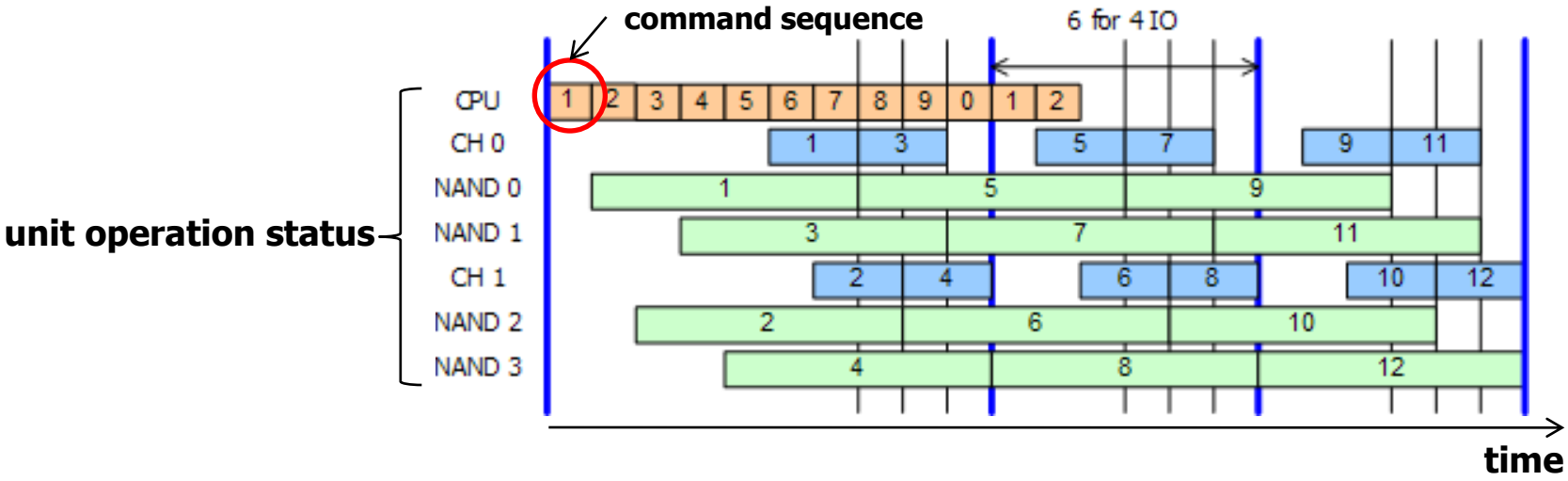
Basic Model (1/5)

- **Assumptions**
  - All resources operate in parallel
  - Firmware runs in non-blocking way
  - Load is evenly distributed on every NAND
  - Repetitive workload - same command is issued infinitely
  - No inter-command dependency
  - IO is aligned with NAND page

Basic Model (2/5)

- **Operation example**
  - **Architecture**
    - 1 CPU
    - 2 channel x 2 bank
  - **Operation**
    - Random read
  - **Busy time in "virtual time unit"**
    - Firmware processing: 1 time unit
    - NAND waiting (tR): 4 time unit
    - DMA transfer (tDMA == tRE): 2 time unit





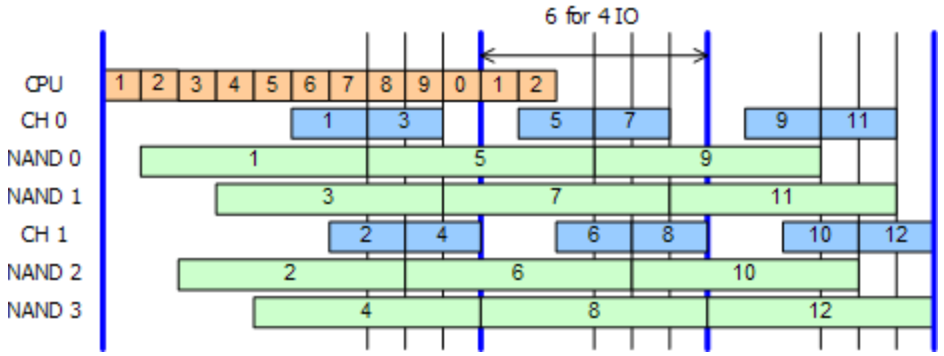
Basic Model (3/5)

- **Observation 1**
  - **Same pattern is repeated except some initial commands**
  - **Periodic model**

- **Latency can be expressed as**
  - ***n*: number of commands in a period**
  - ***t*: length of a period (in time)**
  - ***T(x)*: latency to complete *x* commands**
    - assumes *x* is multiple of *n*

$$T(x) = \frac{x}{n} \times t - t + T(n)$$

$$= \left(\frac{x}{n} - 1 \right) \times t + T(n)$$



Basic Model (4/5)

- **Average latency**
  - **For single IO**

$$\frac{T(x)}{x} = \frac{t}{n} - \frac{t - T(n)}{x}$$

  - **If *x* goes to infinity,**

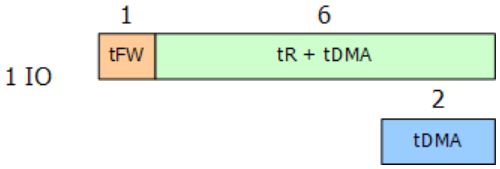
$$\lim_{x \to \infty} \frac{T(x)}{x} = \frac{t}{n}$$

- **Throughput**
  - **IOPS = 1/(average IO latency) = *n/t***

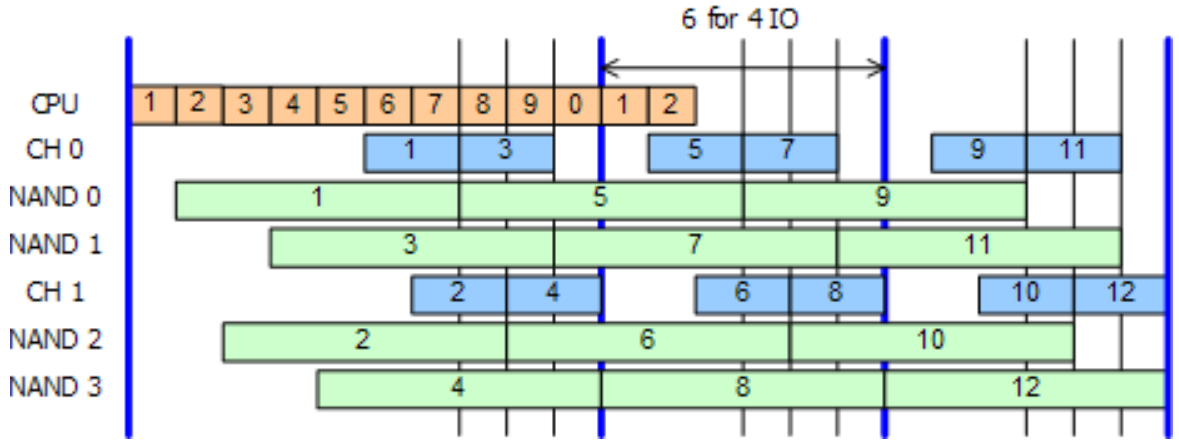
- **How to find *n* and *t* ?**

Basic Model (5/5)

- **Observation 2**
  - **Period is determined by bottleneck resource**
  - **Bottleneck resource can be determined by normalized busy time**
  - ***t* = t_k, *n* = n_k such that t_k/n_k = MAX(t_1/n_1, t_2/n_2, …, t_N/n_N)**
    - n_i: number of resource *i*
    - t_i: busy time of resource *i*
    - *N*: number of resource types
    - *n*: number of commands in a period
    - *t*: length of a period (in time)



MAX(1/1, 6/4, 2/2) = 6/4



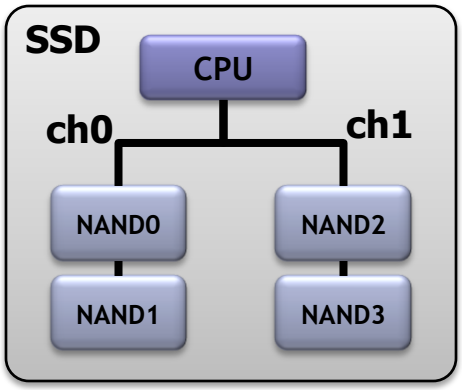
Basic Model - Summary

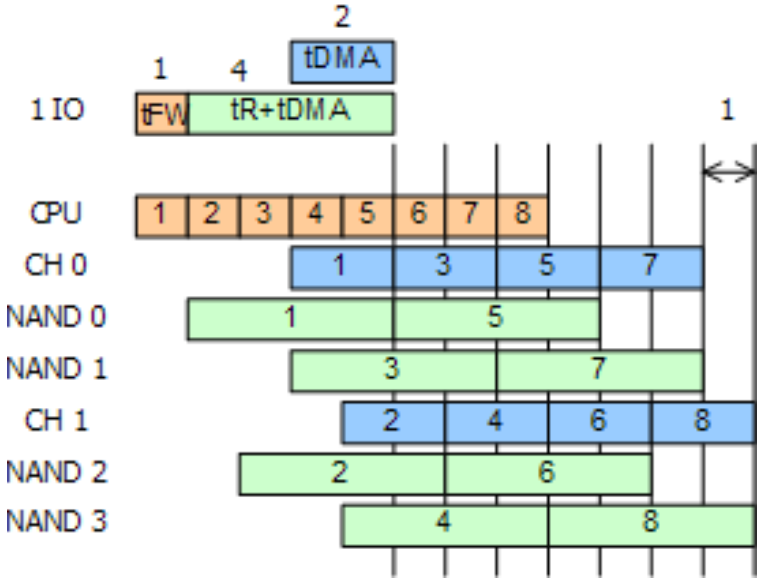
- **Performance model**
  - **given**
    - n_i: number of resource *i*
    - t_i: busy time of resource *i*
    - *N*: number of resource types
  - **tIO = MAX(t_1/n_1, t_2/n_2, …, t_N/n_N)**
    - tIO : average latency to complete one command

- **Useful for exploring performance of SSD**
  - **What if tR or tDMA is changed?**
  - **What's the ideal throughput?**
  - **What if controller gets faster?**

Basic Model – Example (1/4)

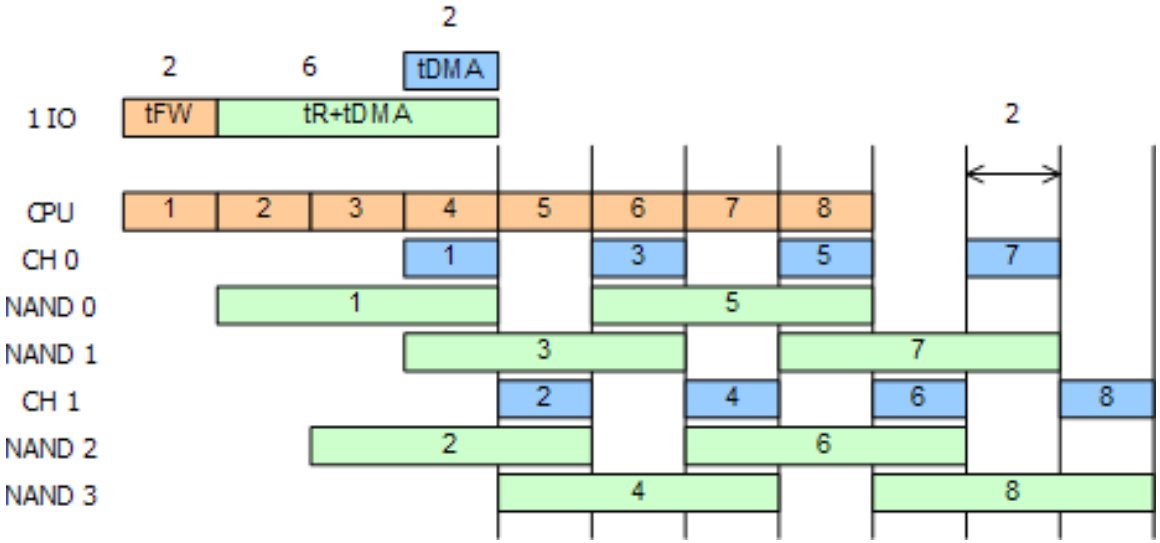
- **Random read**
  - **Assumptions – simple SSD (2ch x 2 bank)**
    - 1 CPU to execute firmware
  - **Performance model**
    - tIO = MAX(tFW/1, (tR+tDMA)/4, tDMA/2)
  - **Equilibrium (optimal) case example**
    - tFW = 1, tR = 2, tDMA = 2
    - tIO = MAX(1/1, 4/4, 2/2) = 1





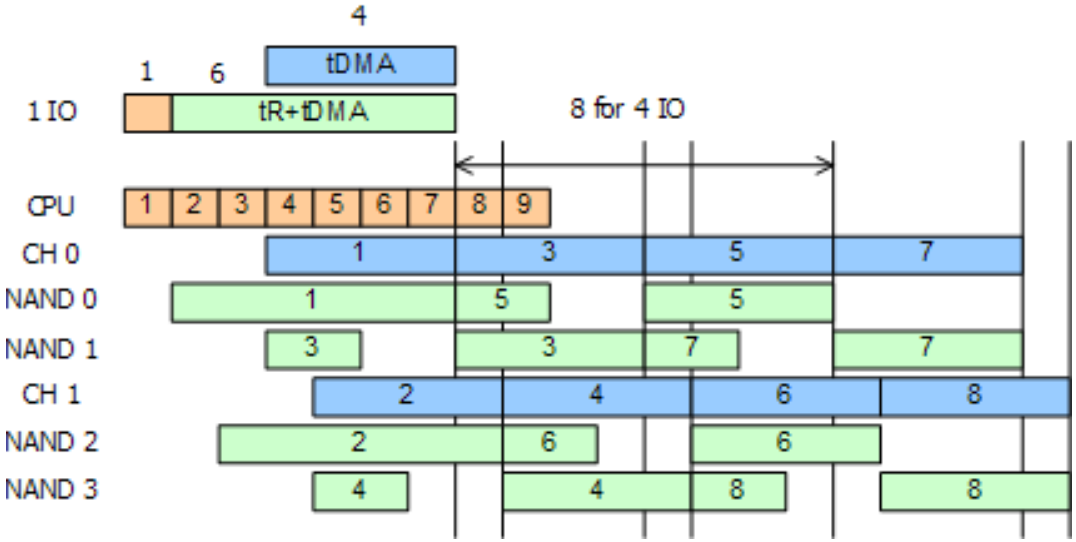
Basic Model – Example (2/4)

- **Random read – case1. firmware bound**
  - **tFW = 2, tR = 4, tDMA = 2**
  - **tIO = MAX(2/1, (4+2)/4, 2/2) = 2**



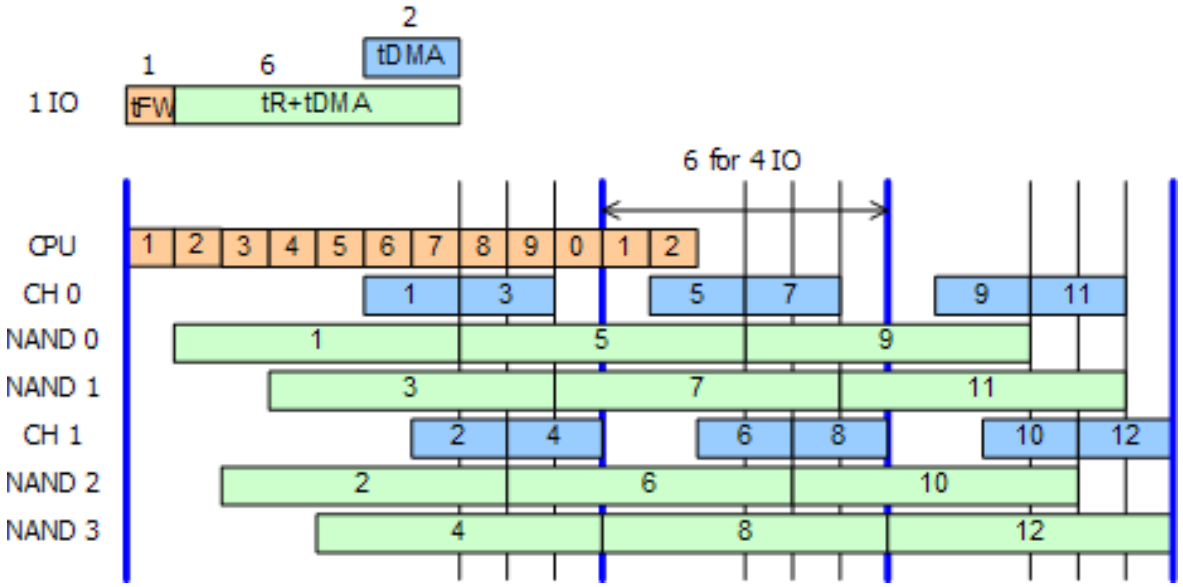
Basic Model – Example (3/4)

- **Random read – case2. DMA bound**
  - **tFW = 1, tR = 2, tDMA = 4**
  - **tIO = MAX(1/1, (2+4)/4, 4/2) = 2**



Basic Model – Example (4/4)

- **Random read – case3. NAND bound**
  - **tFW = 1, tR = 4, tDMA = 2**
  - **tIO = MAX(1/1, (4+2)/4, 2/2) = 1.5**



Advanced Model – Adding Host

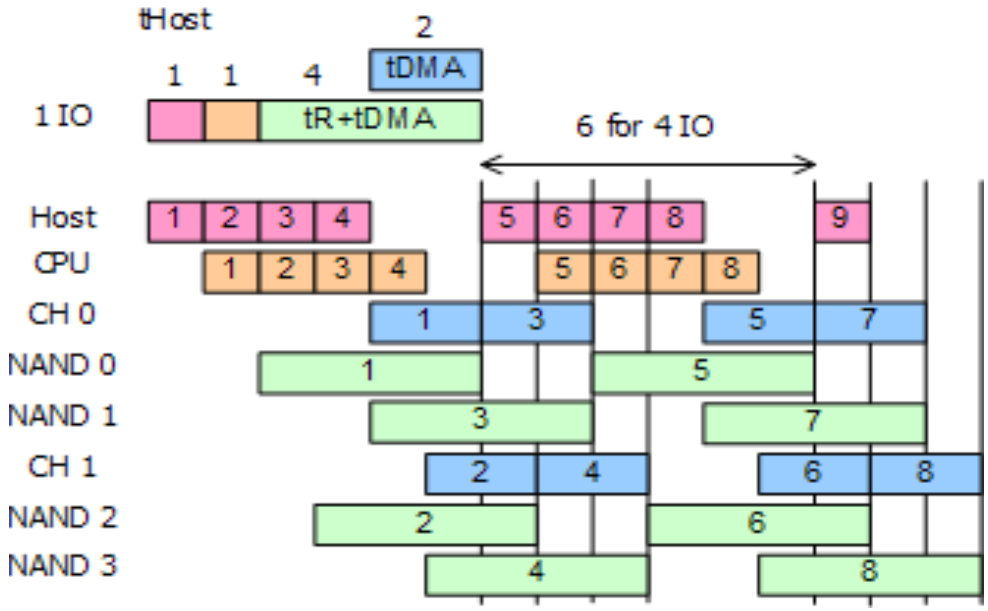
- **Applying the host delay between requests**
  - **Host can be regarded as one kind of resource**
  - **tIO = MAX(tHost, tFW , (tR + tDMA)/4, tDMA/2)**

- **Applying command queuing - finite IO queue**
  - **Incoming IO queue can be regarded as a (virtual) resource**
    - t_Q: required time to complete one IO (= tHost + tFW + tR + tDMA)
    - n_Q: size of incoming IO queue
  - **tIO = MAX(tHost, tFW , (tR + tDMA)/4, tDMA/2, t_Q/n_Q)**
  - **Rationale**
    - Each entry in the queue is in use at least for t_Q time unit.
    - A new IO request can be queued only when there exists an empty entry.

Advanced Model – Example

- **Command queue size = 4**
  - **tIO = MAX(tHost, tFW , (tR + tDMA)/4, tDMA/2, t_Q/4)**
  - **tHost = 1, tFW = 1, tR = 2, tDMA =2**

- **Then,**
  - **tIO = MAX(1, 1, (2+2)/4, 2/2, (1+1+2+2)/4) = 6/4 = 1.5**



SSD Performance Models

- **Random read**
  - **As explained**

$$t_{read} = MAX\left(t_{Host}, t_{FW}, \frac{t_R + t_{DMA}}{n_{NAND}}, \frac{t_{DMA}}{n_{Channel}}, \frac{t_{Host} + t_{FW} + t_R + t_{DMA}}{n_{Queue\_depth}} \right)$$

- **Sequential read**
  - ***a*** **- number of pages to read per single read**
  - **T(*a*, …) – time to process single sequential read with *a* pages**

$$t_{read} = MAX\left(t_{Host}, t_{FW}, \frac{a(t_R + t_{DMA})}{n_{NAND}}, \frac{a \times t_{DMA}}{n_{Channel}}, \frac{t_{Host} + T(a, R_{CPU}, R_{NAND}, R_{Channel})}{n_{Queue\_depth}} \right)$$

SSD Performance Models

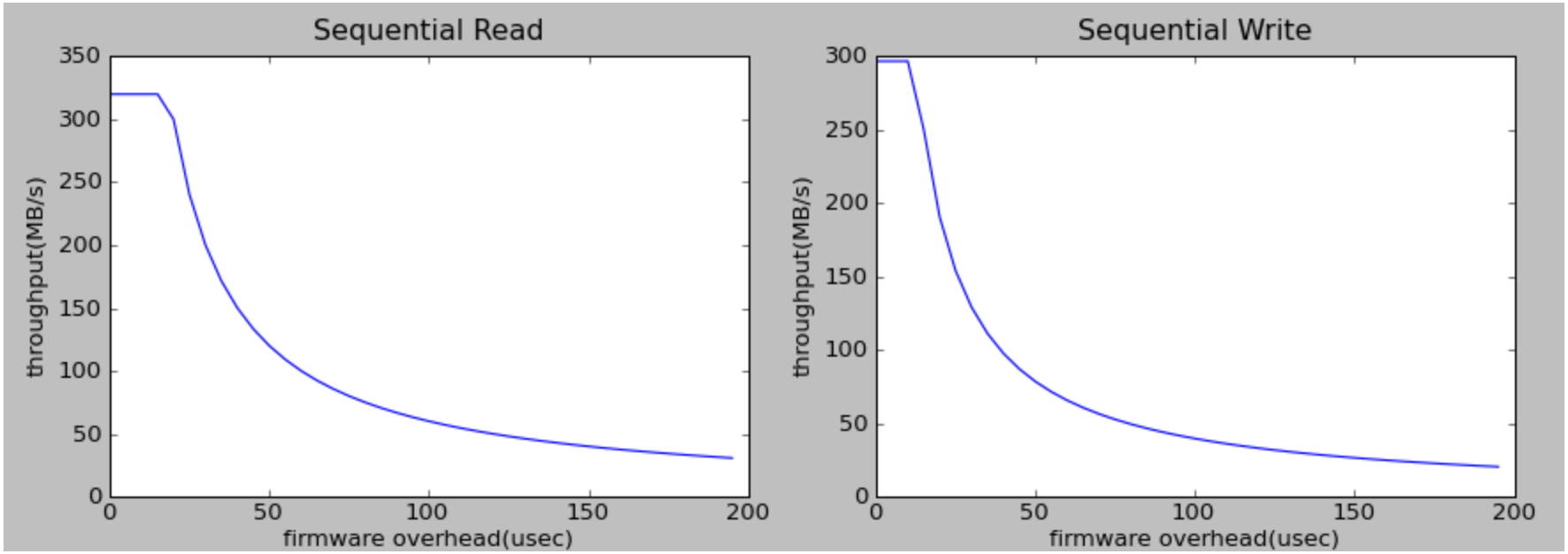
- **Write performance is dependent on mapping**
- **Assumption – page mapping**
  - **Every NAND has at least one free block for merge**
  - **During merge, all write operations will be blocked**
  - **Switch merge for sequential write, full merge for random write**
- **tWrite = tIO + tMerge x Merge_frequency**
  - **tIO calc is similar to read (replace tR with tProg)**
  - **Sequential write (switch merge)**
    - tMerge = tBER
    - Merge_frequency = 1/(pages_in_block x number_of_NAND)
  - **Random write (full merge)**
    - tMerge = 2xtBER + tCopyBack x pages_in_block
    - Merge_frequency = 1/(pages_in_block x number_of_NAND)

Performance Estimation (1/3)

- **Assumptions (or constants)**
  - **8 channel x 8 bank**
  - **tHost = 10us, NCQ = 32**
  - **NAND: large block SLC (x8)**
    - Page size = 2KB, pages in block = 64
    - tR = 20us, tProg = 200us, tBER = 2000us, tRE/tWE = 25ns
- **Variables**
  - **tFW: 0us (ideal) ~ 200us**

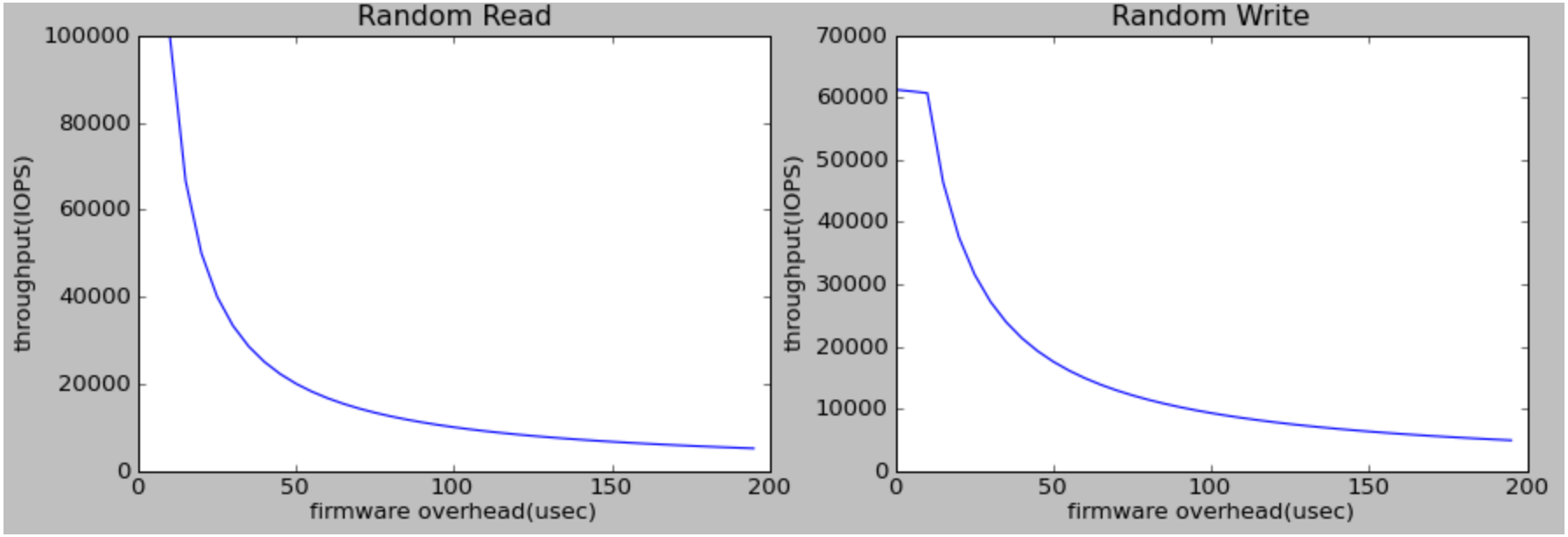
Performance Estimation (2/3)

- **Sequential I/O bandwidth**
  - **Bounded by I/O time (tR/tProg/tDMA)**



Performance Estimation (3/3)

- **Random IOPS**
  - **Bounded by firmware overhead**
  - **Firmware overhead = CPU time + memory access + etc**



Future Work – More Parameters!

- Accuracy of firmware overhead
  - Architecture – CPU clock, multi-core, bus topology, HW acceleration
  - Mapping algorithms - BAST, FAST, …
- NAND flash memory
  - High-speed I/F (ex. ONFI)
  - Copy-back condition (internal, external, R4CB)
  - Cache read/program
- Workload
  - Micro benchmark - Sub-page I/O, Misaligned I/O
  - Synthetic benchmark - PCMark05, SysMark
  - Effect of trim(?)

Conclusion

- We can estimate performance of SSD using analytic modeling
  - Parameters - architecture, NAND, firmware, workload
- Firmware overhead is not negligible in SSD where I/O resources operate in parallel
- Call for action – more sophisticated performance modeling!