ECEN 5053-002

Developing the Industrial Internet of Things

Week 9 - Lecture

Sensors, Drives, File Systems

Dave Sluiter - Spring 2018

Material

- Sensors
- Storage: Hard Drives and Solid State Drives
- File Systems

Learning Outcomes

- Thermistors
  - How to calibrate
  - How to filter
  - How to sample
- Understand how block and object drives function
- Understand how hard drives and solid state drives operate
- Grasp how traditional file systems fail to meet the needs of large data sets
- Gain initial insight into file systems:
  - NFS
  - Hadoop
  - Lustre

Sensors

- One of the key skills to acquire is understanding sensors: how to calibrate them, and the filtering required.
- We will look at temperature sensors
  - Thermocouples
  - Resistive Temperature Devices (RDT)
  - **Thermistors**

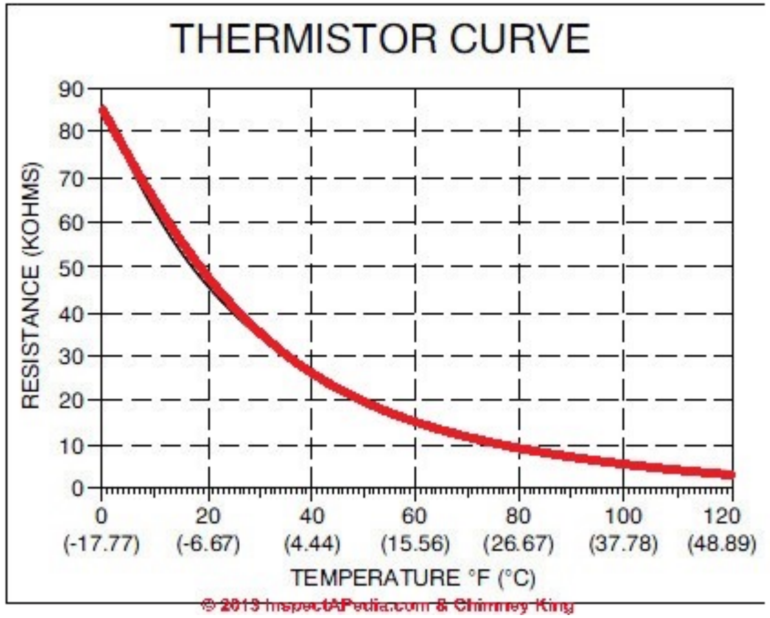
Thermistor

- Is a 2-wire device that changes resistance as a function of temperature
- For example:
  - -40C to +200C
  - 220K to 10K ohms

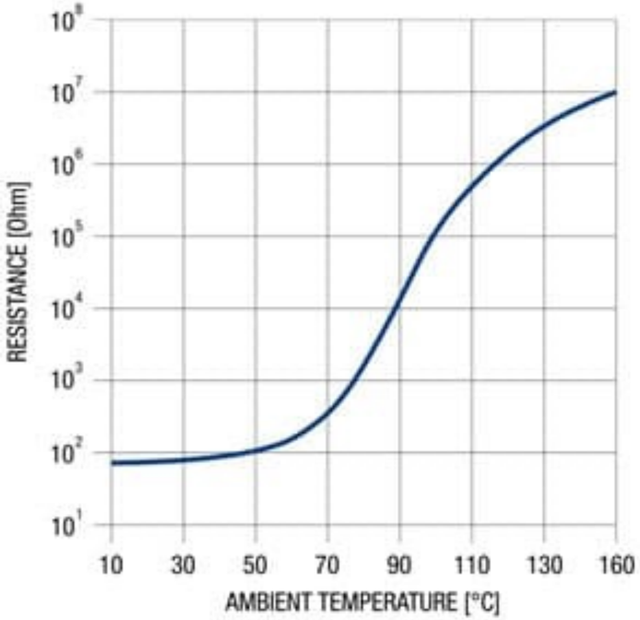
Temperature Coefficient

- Some have negative temperature coefficients
- Some have positive temperature coefficients

NTC



PTC



Terminology

- Before we get into this, we have to understand the terms:
  - Resolution
  - Precision
  - Accuracy
  - Tolerance

Resolution

- Is the “**fineness**” to which an instrument (sensor) can be read
  - Analog stopwatch can be read to 1/10 of a second
  - Digital stopwatch can be read to 1/100 of a second

Source: http://www.tutelman.com/golf/measure/precision.php

Precision

- Humans take about 1/10 of a second to respond to stimulus
  - So this means that the digital stopwatch, even though it has 1/100 second **resolution**, only has 1/10 second **precision**. What?
- **Precision**: How ***repeatable*** a measurement is
- Because of the human factor, the digital stopwatch is only repeatable to 1/10 second. Testing would show that the digit in the 1/100 place would be close to random.

Accuracy (correctness)

Improved precision (repeatability)
Accuracy remained the same,
average is still just as far away from
the center.

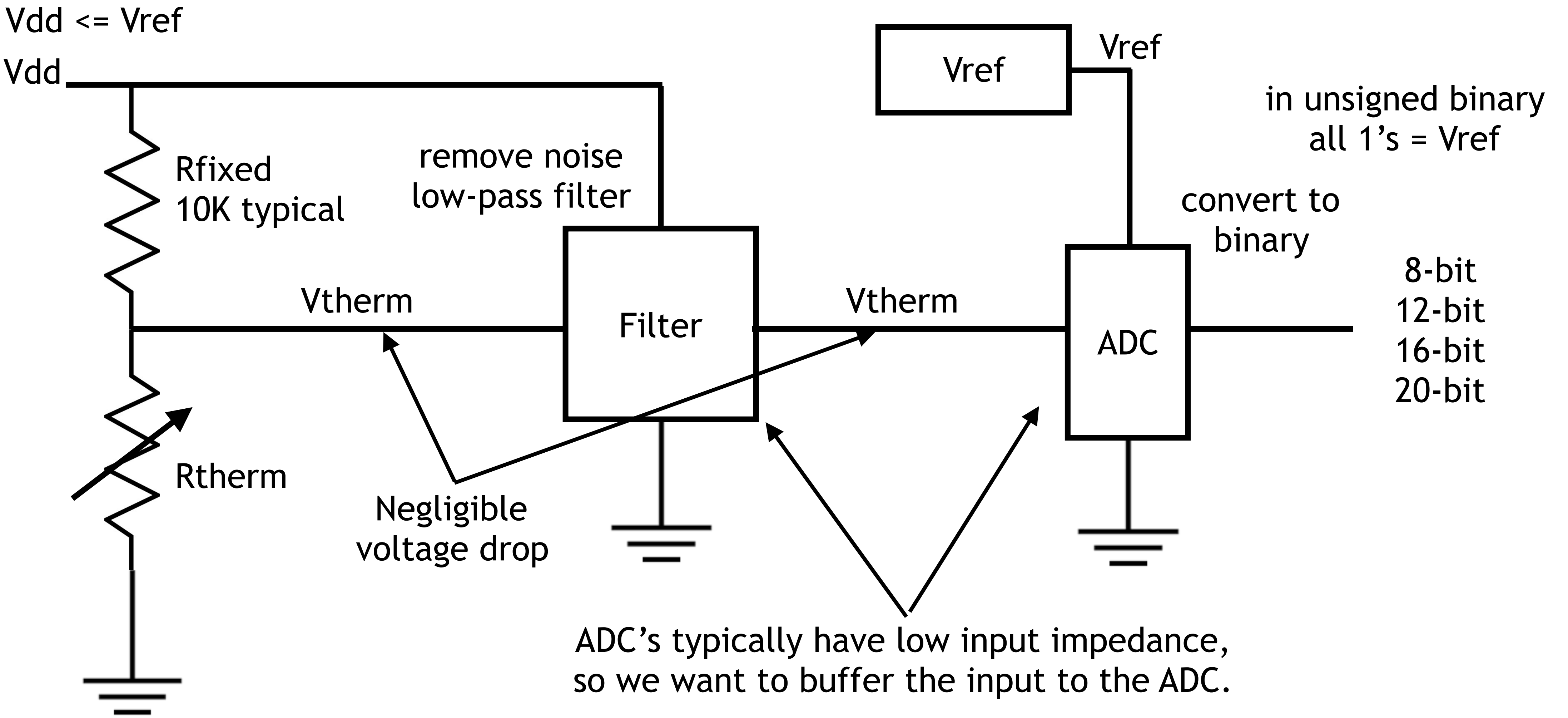
Improved accuracy with
the same precision.

Improved accuracy +
improved precision.

Tolerance

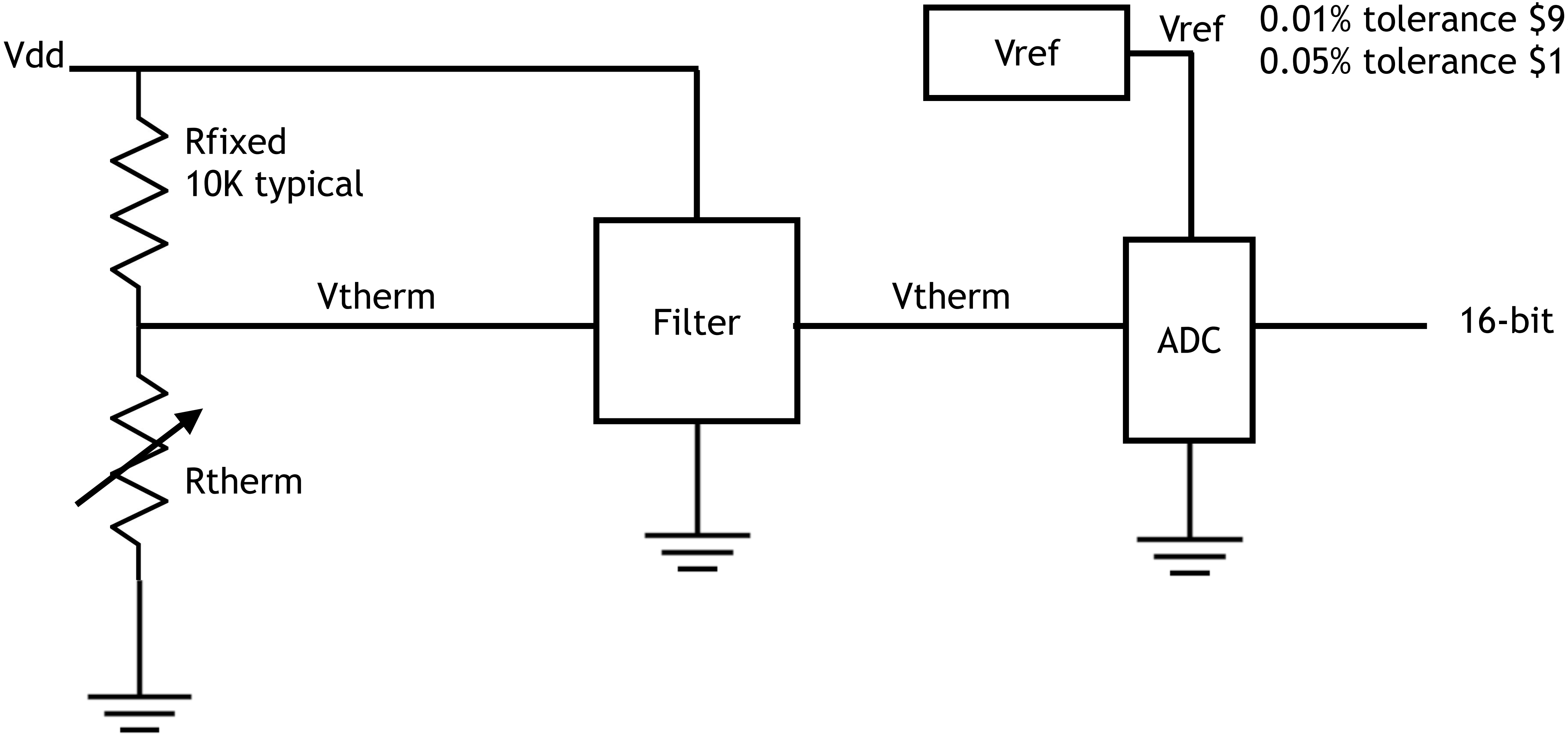
- Combines precision (repeatability) and accuracy (correctness)

Basic Sensor Circuit



ADC's typically have low input impedance,
so we want to buffer the input to the ADC.

Basic Sensor Circuit



0.01% tolerance $93
0.05% tolerance $15

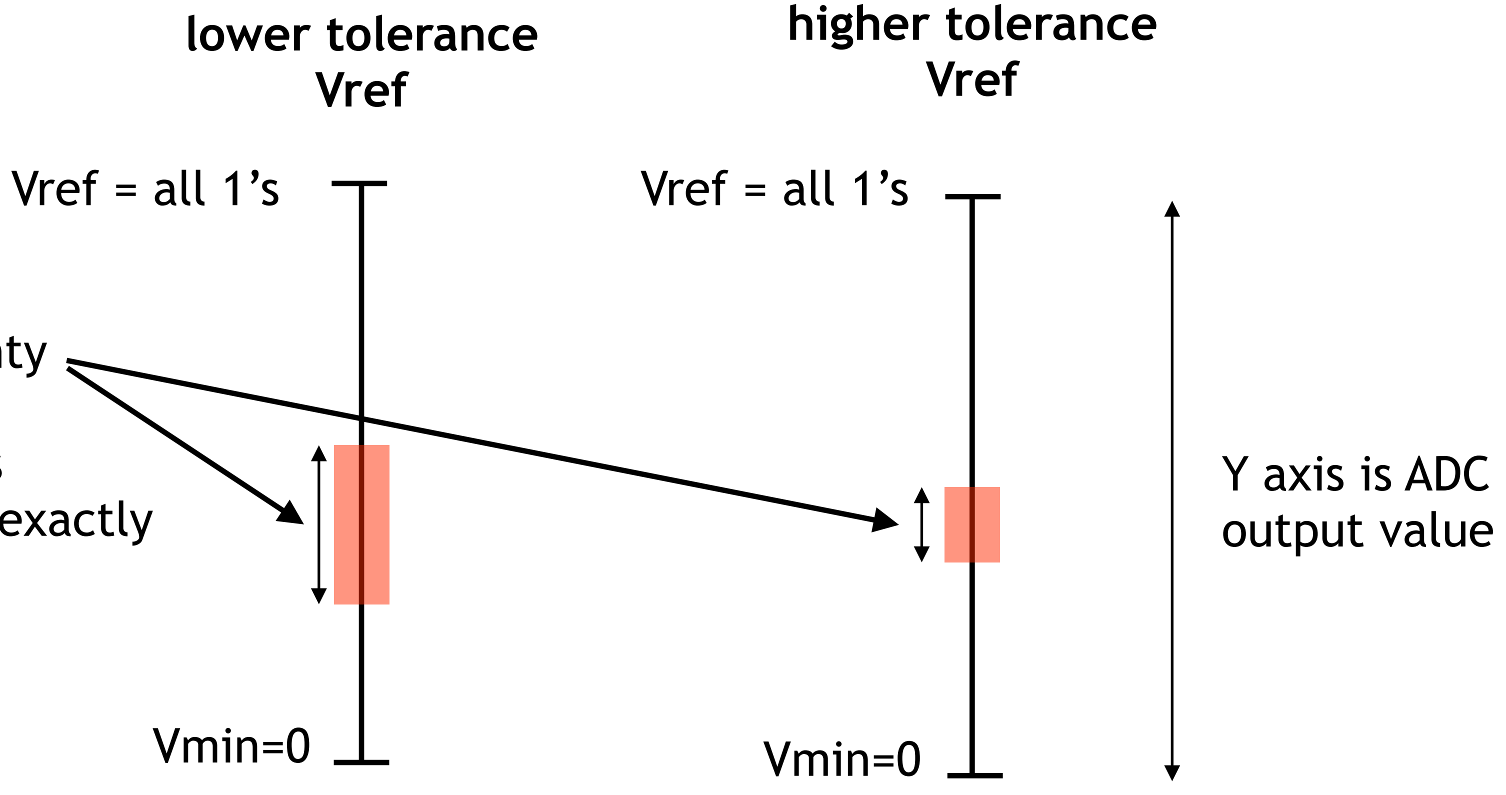
Accuracy

- So say we had a 16-bit ADC (unsigned)
- and a 0.01% tolerance Vref part
- What is the accuracy of the ADC?
  - (2^16) * 0.0001 = 6.554
- This means any measurement we take is +/- 3.277, say +/- 3.3 units from the actual (correct) voltage
- If we wanted to save money and substituted a 0.2% tolerance Vref part for $0.40, we get
  - (2^16) * 0.002 = 131 or +/- 65.5 units from the actual (correct) voltage

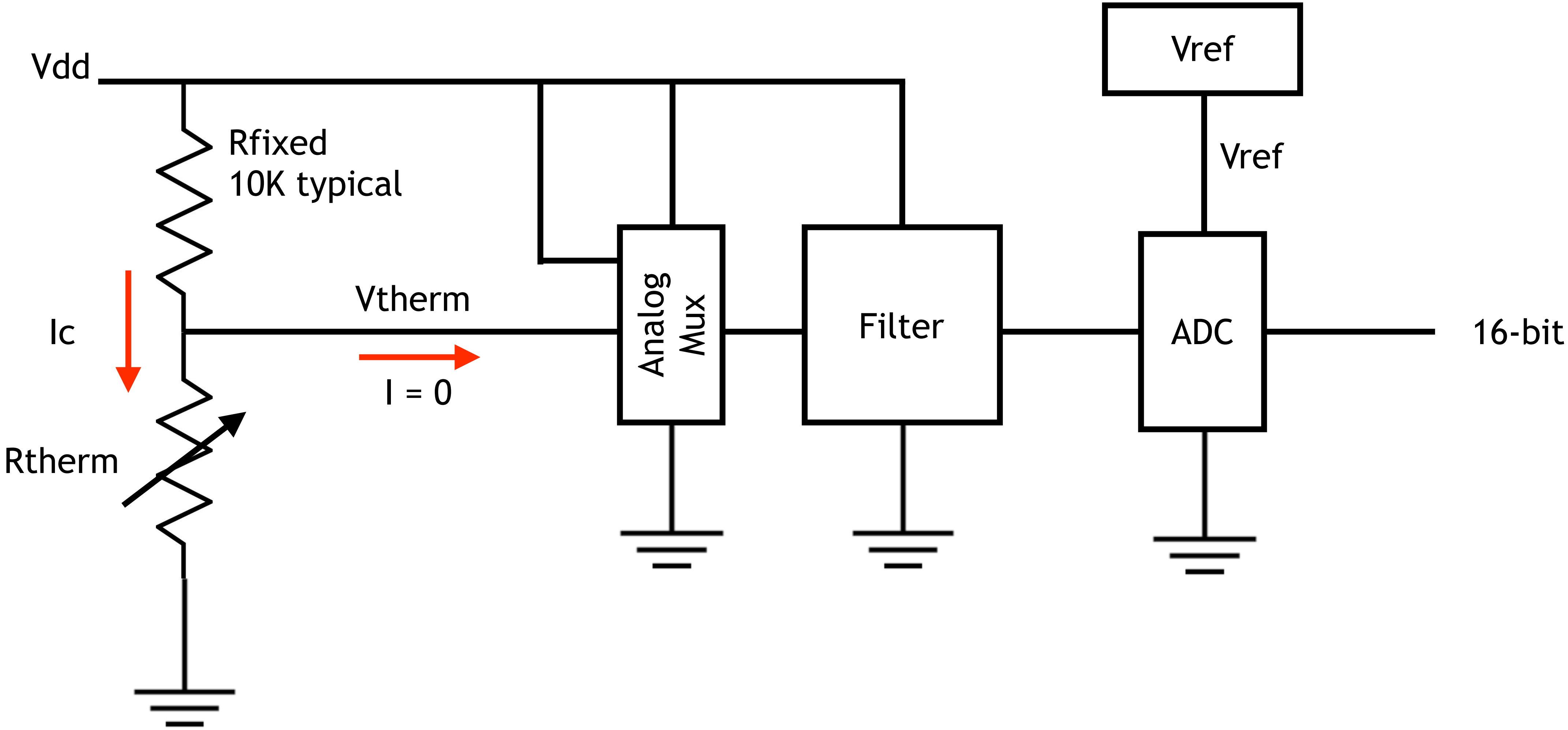
Accuracy

Any given sample, region of uncertainty

We know the actual value is in this region, we just don't know where exactly



Advanced Sensor Circuit



Calculating Rtherm

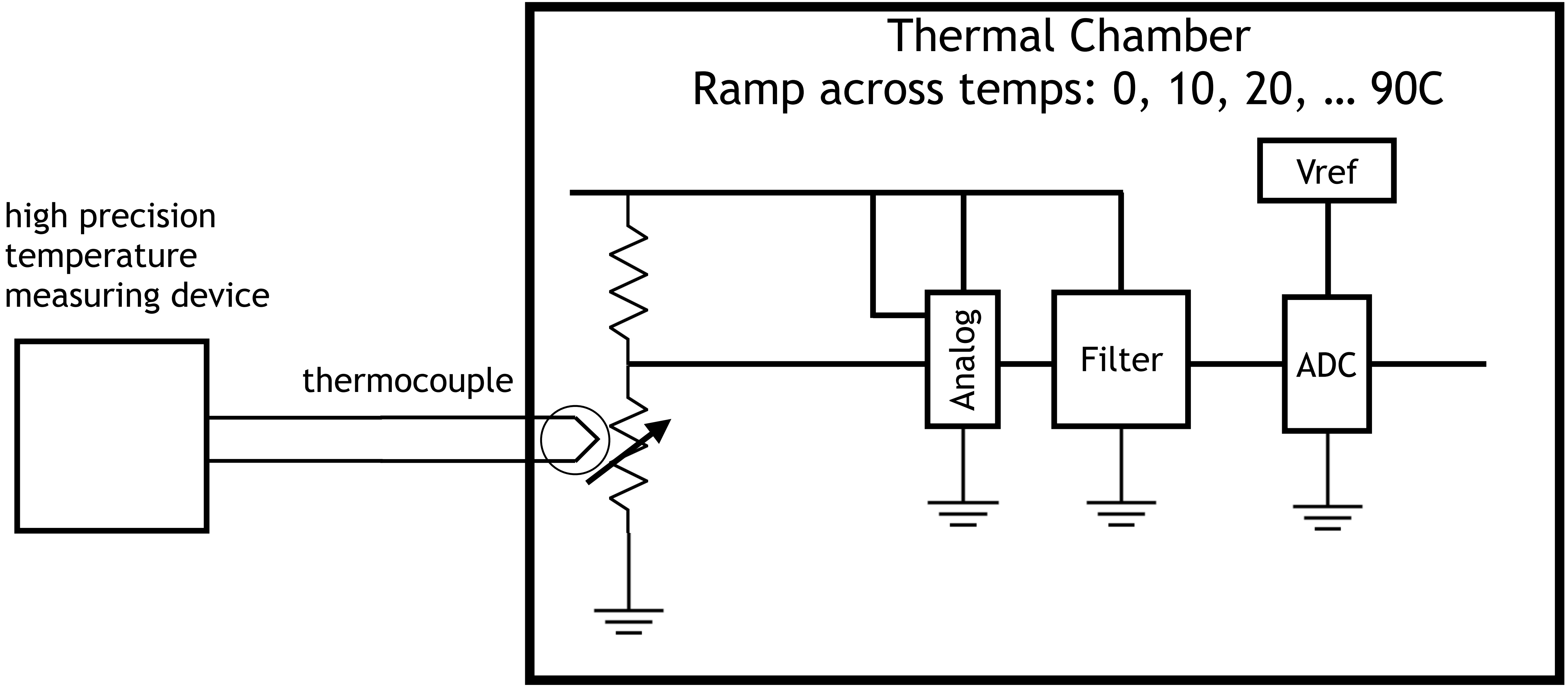
- Measure Vdd = MVdd
- Measure Vtherm = MVtherm
- Compute current Ic = (MVdd - MVtherm) / Rfixed
- Ic = MVdd / (Rfixed + Rtherm), solving for Rtherm
  - Rtherm = (MVdd/Ic) - Rfixed
- Rtherm is looked up in the manufacturer's data sheet (a table)
  - produce a temperature value

Note: A divide operation is required. So your CPU needs a divide instruction or you can use a software library.

Calculating Rtherm (con't)

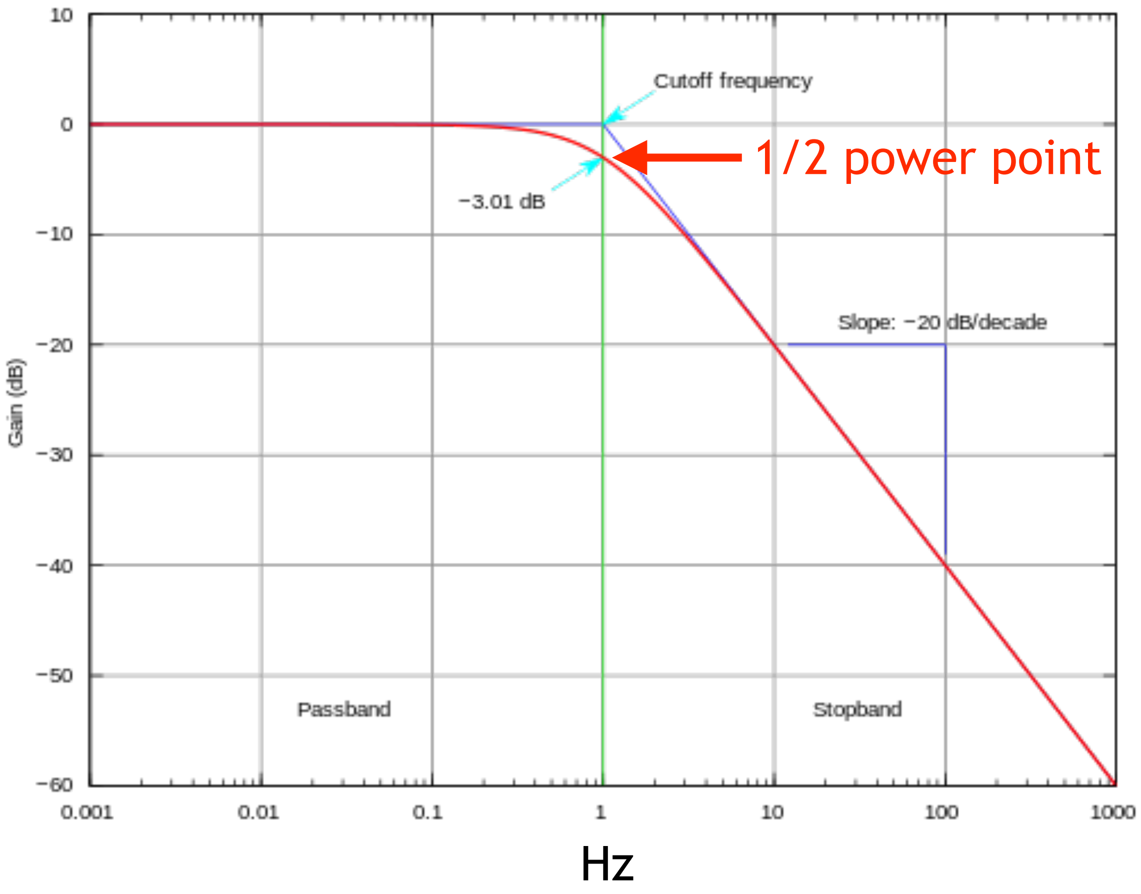
- What about tolerances T?
- Measure Vdd = MVdd + T
- Measure Vtherm = MVtherm + T
- Compute (MVdd + T) - (MVtherm + T)
  - the tolerance term cancels out, and we have:
  - (MVdd - MVtherm)

Validating Temperature Readings

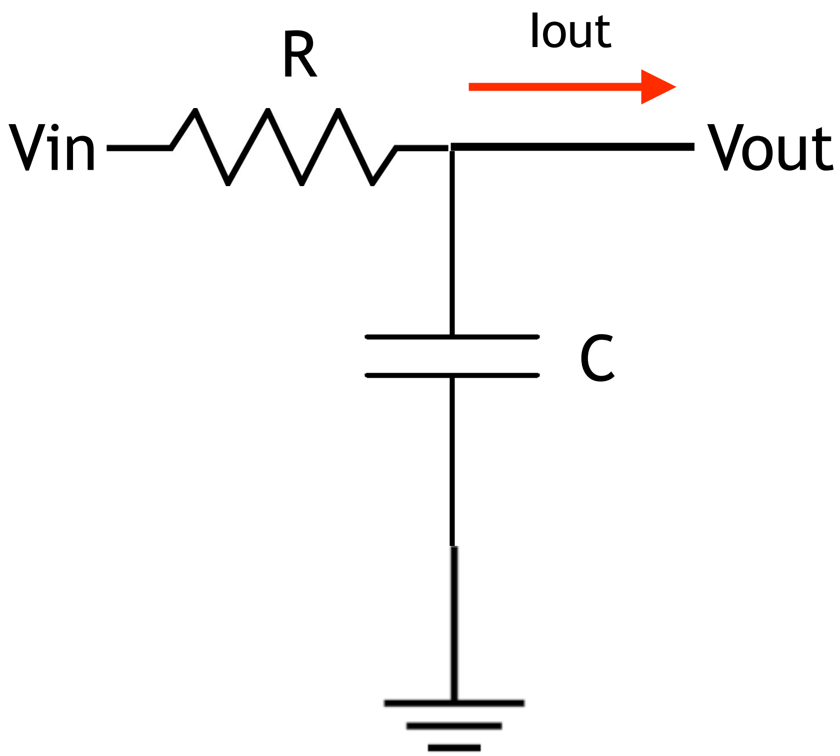


Filtering

- Some applications require filtering out as much noise as possible from our ADC samples
- Pass Vtherm through a low-pass filter
  - F_C = Cutoff frequency is the -3 dB down point
  - Divides the passband from the stopband



Simple Passive Filter



Cutoff frequency is given by:
fc = 1 / (2*pi*R*C) Hz

We choose:
R = 100 ohms
fc = 1 Khz

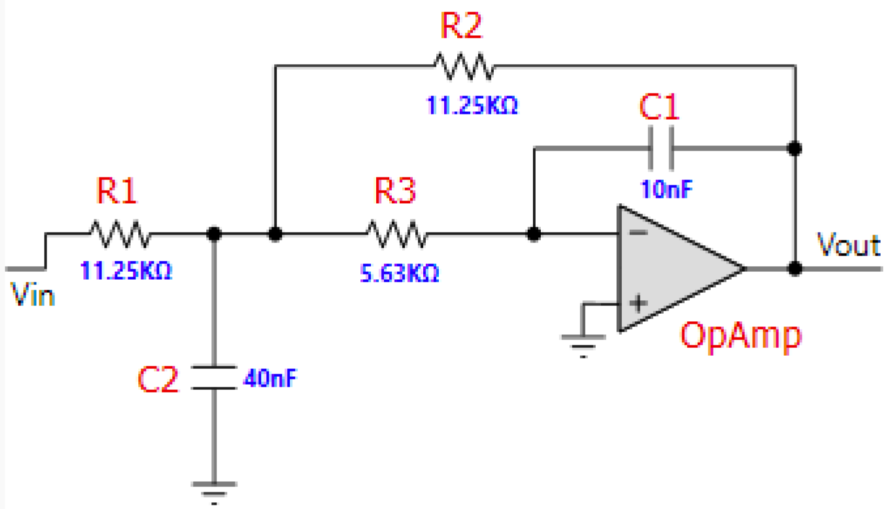
1000 = 1 /2 *pi*R*C
C*R*pi*2 = 1/1000
C = 1/1000*R*pi*2
C = 1.6 uF

Vout = Vin in steady state, Iout = 0

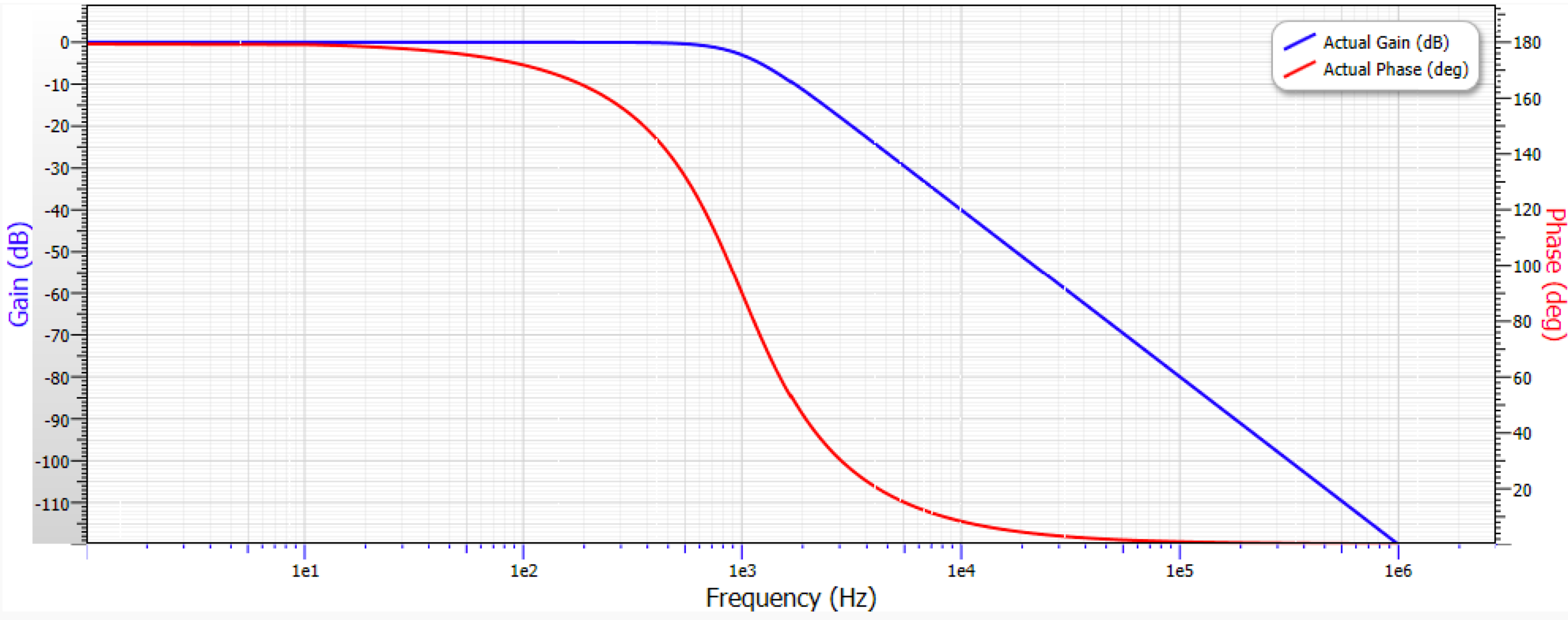
Also doesn't provide any buffering

Simple Active Filter

Gain = 1 Vout/Vin



Filter Stage:	1
Passband Gain(Ao) :	1
Cutoff Frequency(fn):	1 kHz
QualityFactor (Q):	0.71
Filter Response:	Butterworth
Circuit Topology:	MultipleFeedback
Min GBW reqd.:	71 kHz

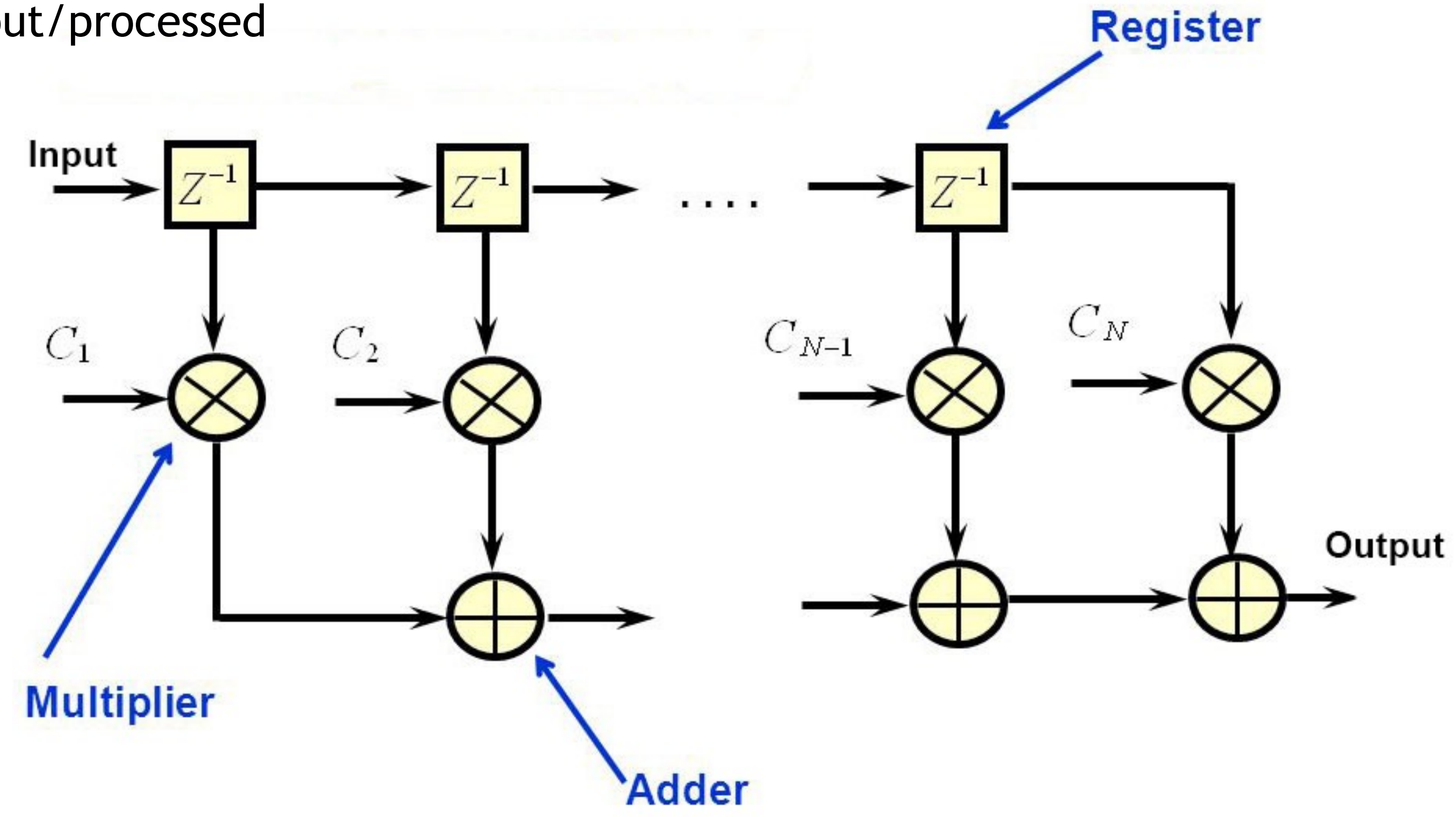


Source: Filter Pro Desktop: http://www.ti.com/lsds/ti/analog/webench/webench-filters.page

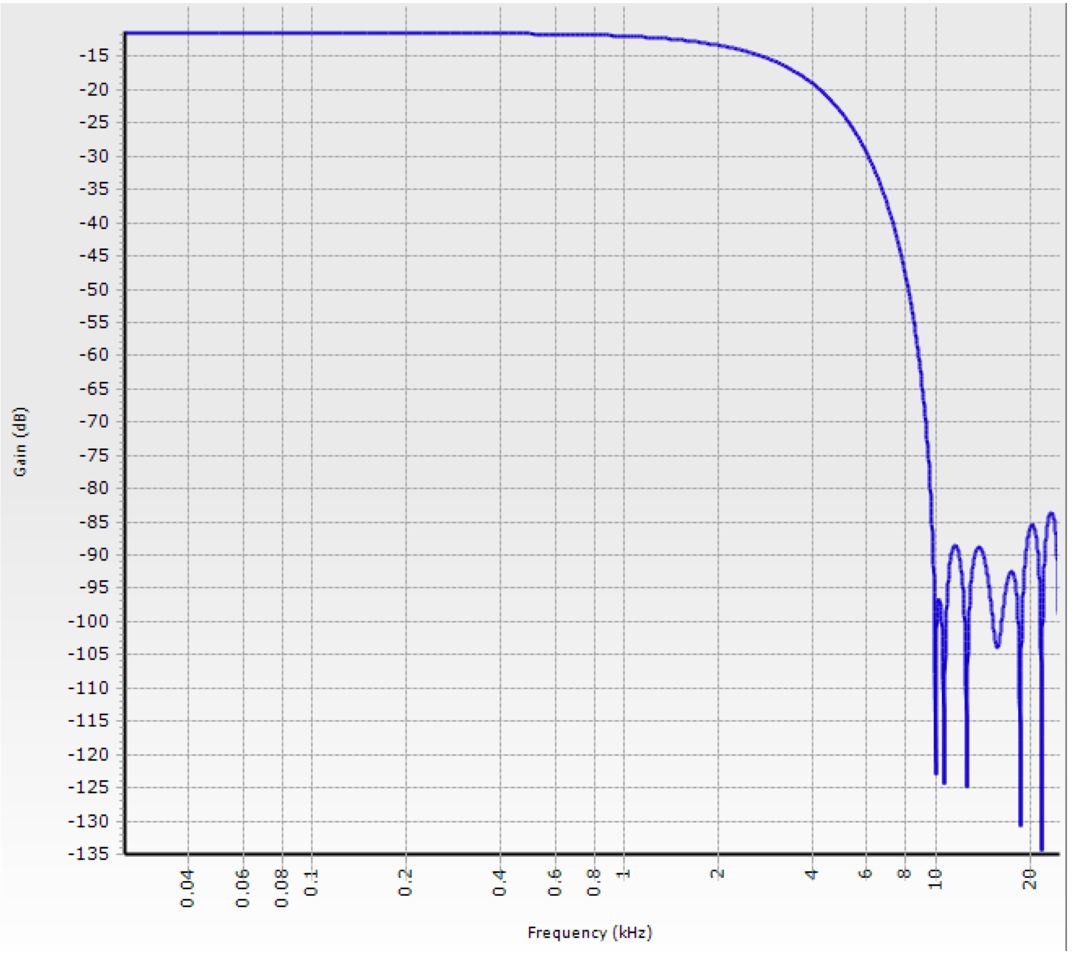
Digital FIR Filter

- Finite Impulse Response Filter

Note: There is a clock, samples are input/processed at the filter sample frequency, FF_s

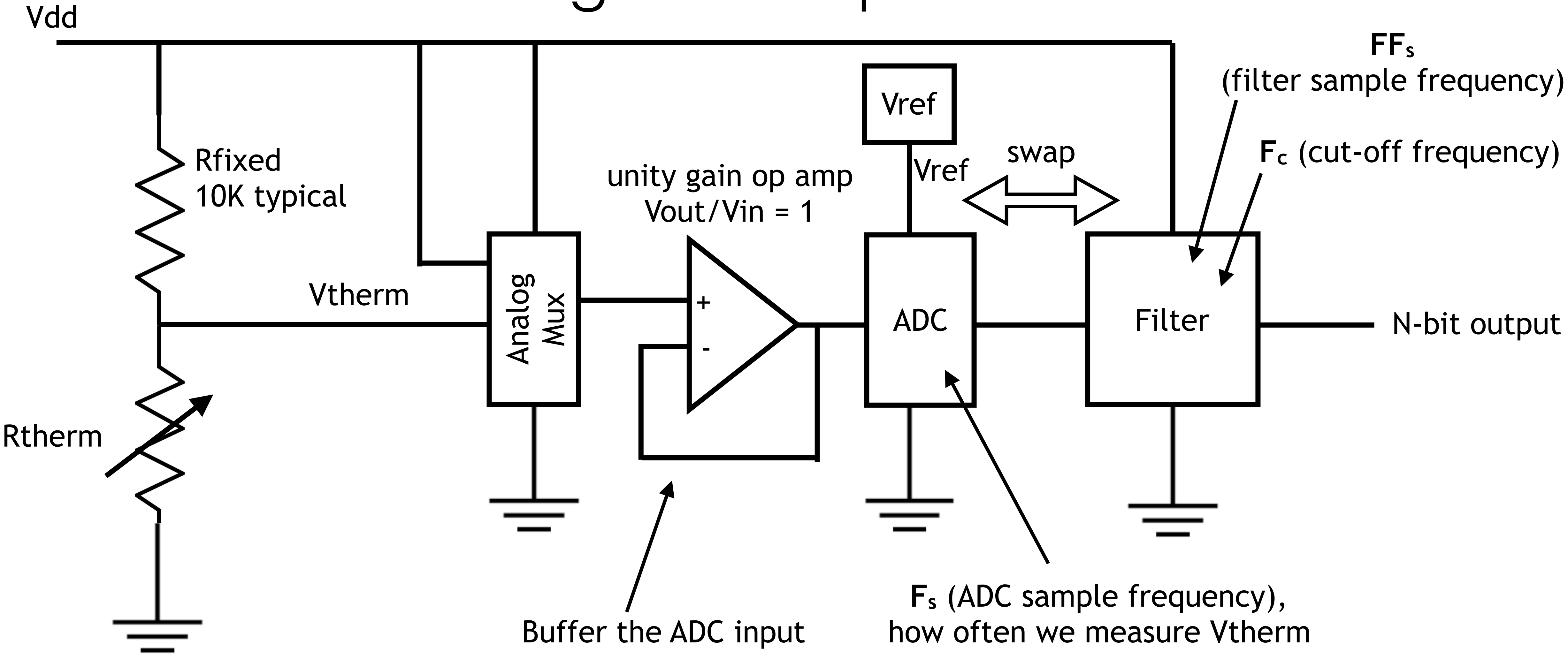


Digital FIR Filter Response

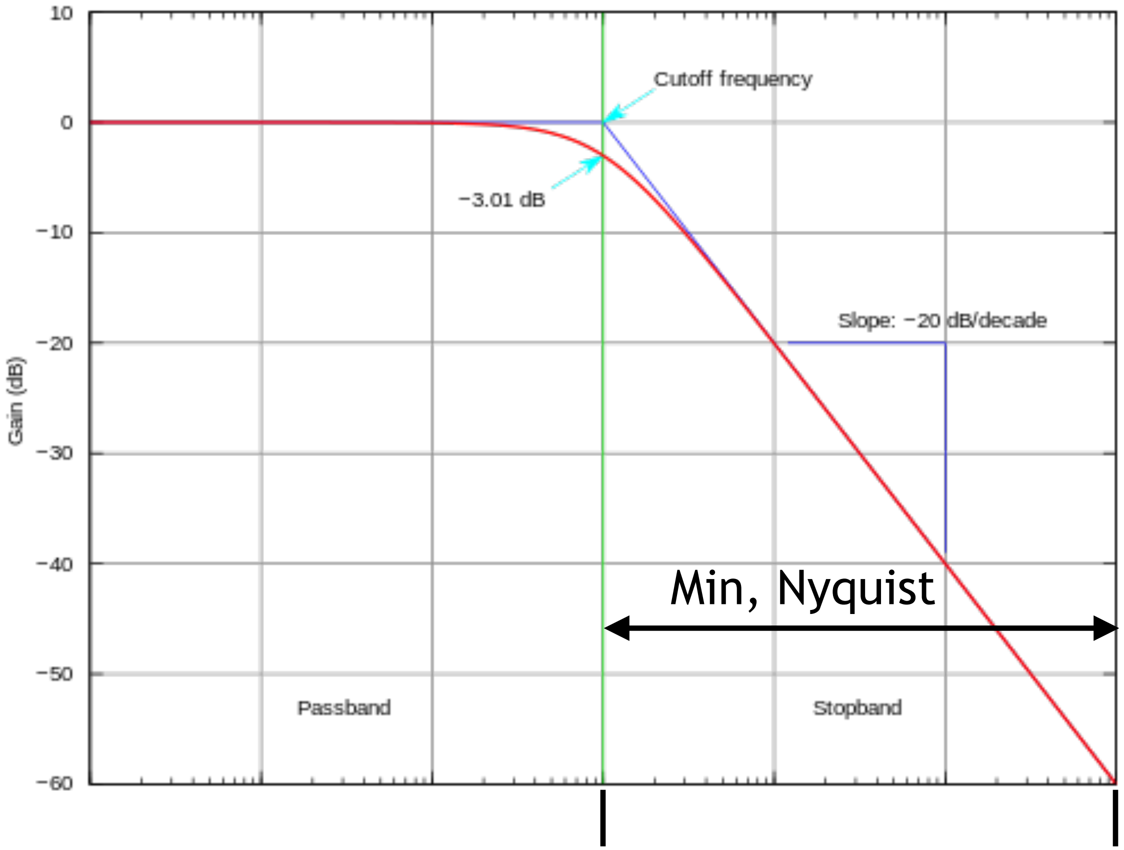


Source: http://www.cypress.com/forum/psoc-creator-software

But wait, doesn't the FIR filter operate on digital samples?

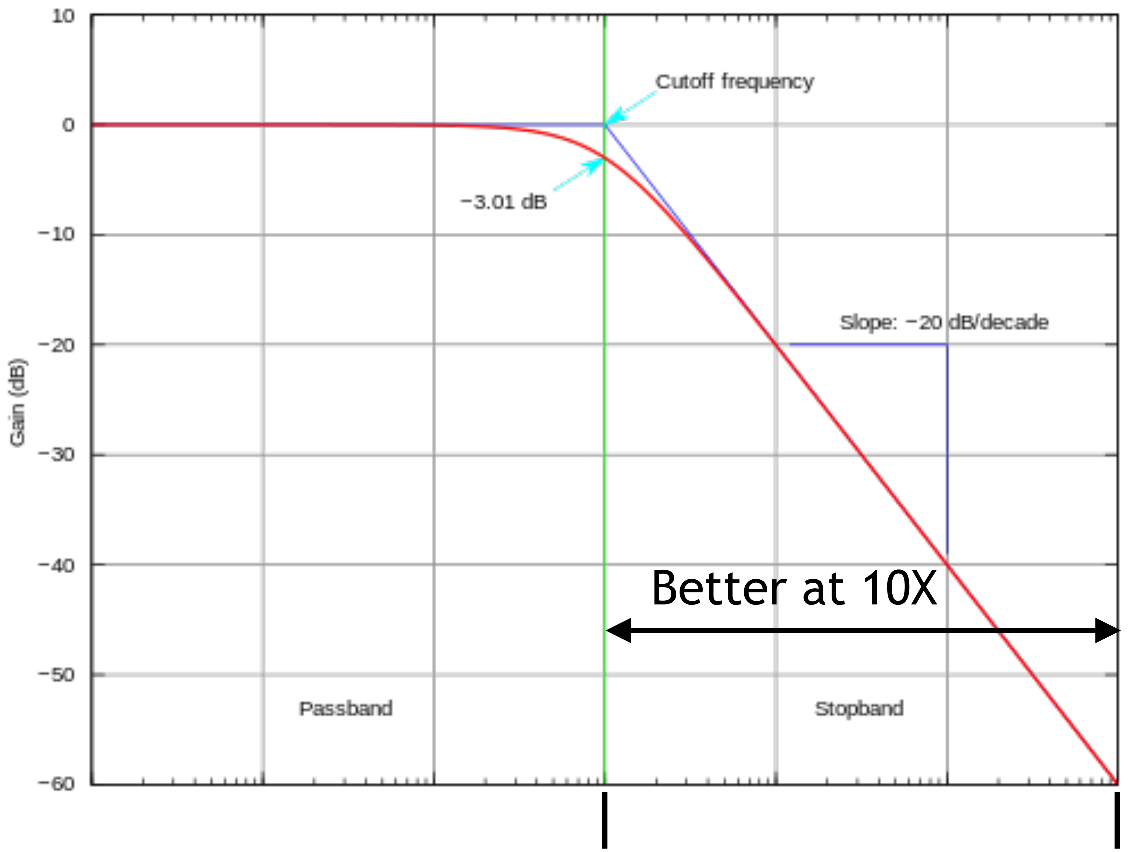


What should these frequencies be?



F_C= 500 Hz

F_S = FF_S = 1 KHz



F_C= 500 Hz

F_S = FF_S = 5 KHz

Demos

- TI FilterPro
- Cypress PSoC Creator

Top Sensor Manufacturers in 2017

- STMicroelectronics (Switzerland),
- NXP Semiconductors N.V. (Netherlands),
- TE Connectivity Ltd. (U.S.),
- Infineon Technologies AG (Germany),
- Texas Instruments Incorporated (U.S.),
- Robert Bosch GmbH (Germany),
- Analog Devices, Inc. (U.S.),
- ams AG (Austria),
- Honeywell International, Inc. (U.S.),
- Sensirion AG (Switzerland),
- Knowles Electronics, LLC. (U.S.),
- InvenSense, Inc. (U.S.),
- Omron Corporation (Japan),
- ARM Holdings Plc. (U.K.),
- ABB Ltd. (Switzerland),
- Emerson Electric Company (U.S.),
- Siemens AG (Germany), Broadcom Limited (U.S.), and
- Asahi Kasei Corporation (Japan).

Sensor Summary

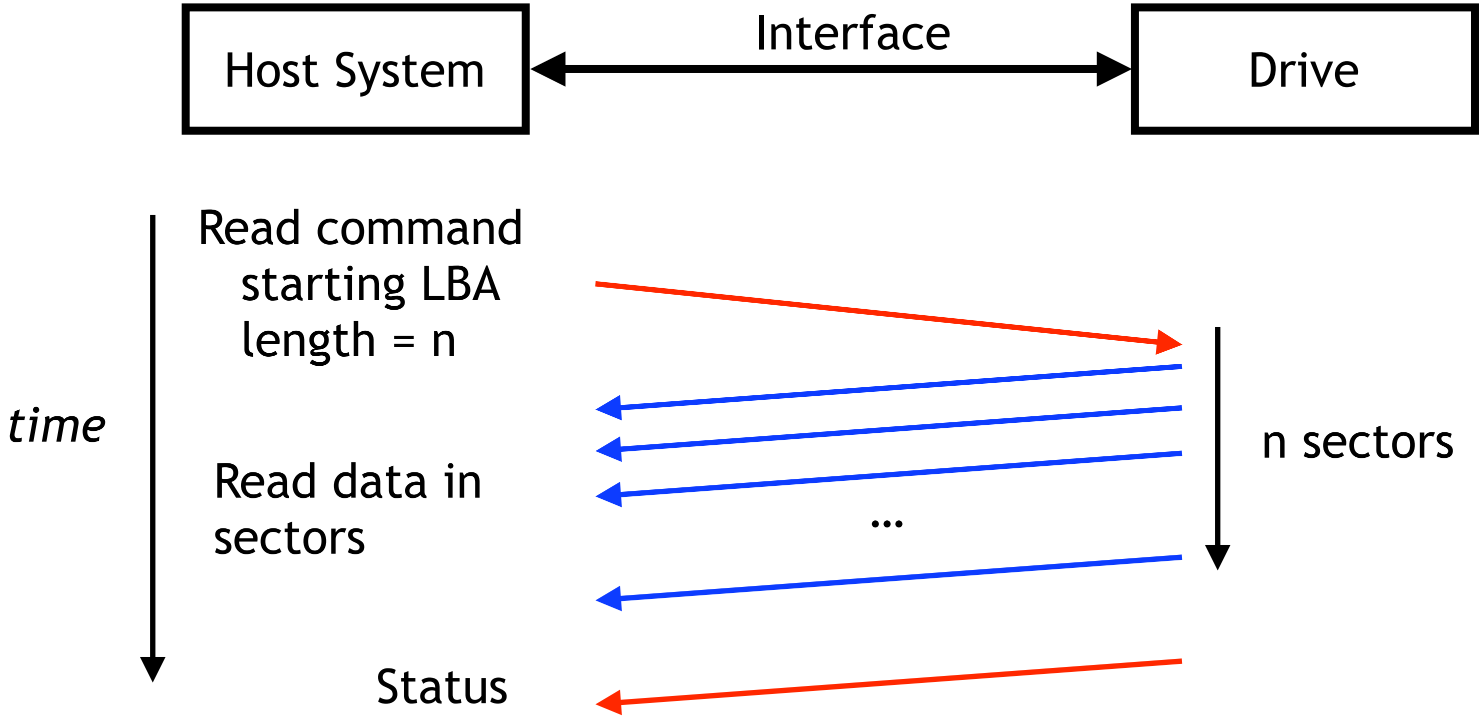
- Goals
  - Saw examples of how to calibrate, filter, sample and validate temperature sensors
    - Remember: This was a key skill our Industry Advisory Board wants to see in our ESE graduates
  - For more in-depth learning take Jimmy Zweighaft’s course “*Embedding Sensors and Actuators*”

File Systems

- Block based vs. Object based drives
- File Systems:
  - NFS
  - Apache Hadoop (HDFS)
  - Lustre (LFS)
- Others (not covered)
  - *ZFS*
  - *Quantcast*
  - *Ceph*
  - *Gluster*
  - *PVFS*

Block Devices

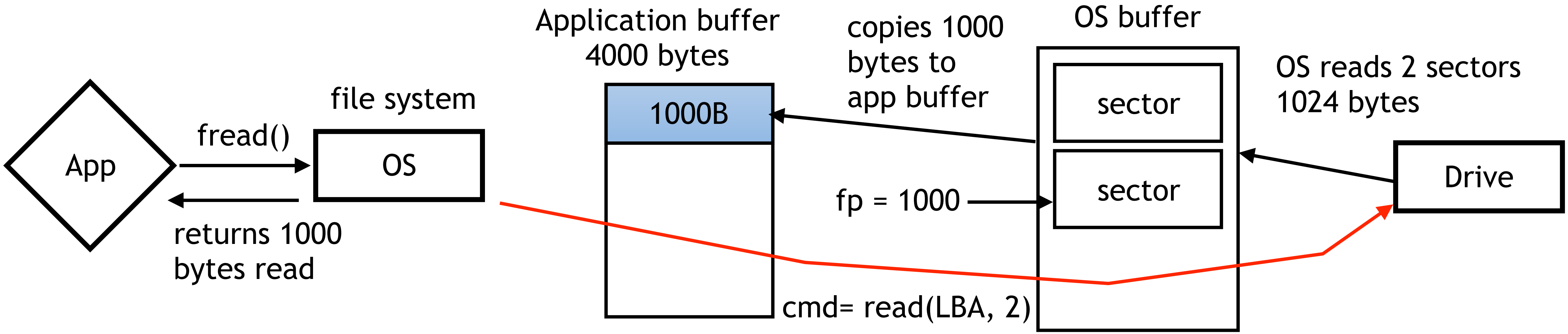
SATA - Serial ATA
SAS - Serial Attached SCSI
USB - Universal Serial Bus
PCIe / NVMe



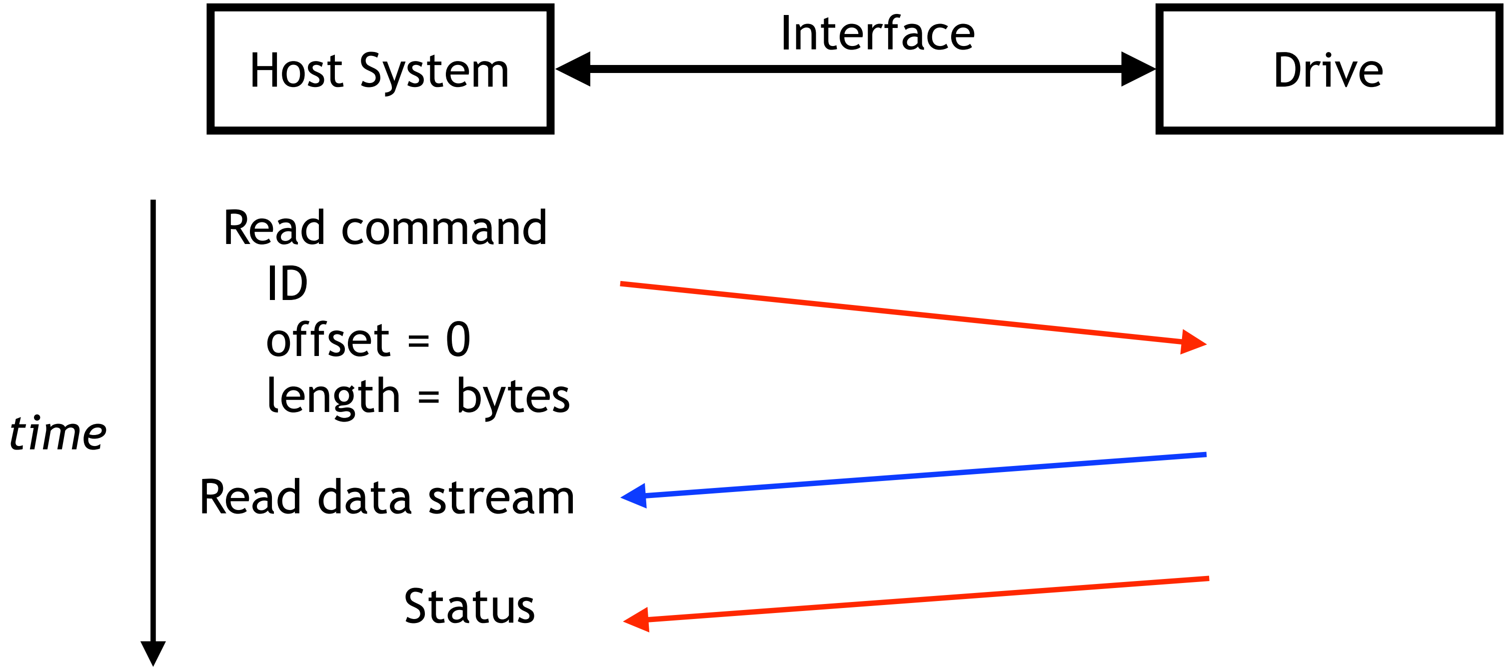
* Drives store data in **blocks**, also called **sectors**.
* Each block has an address, referred to as the **Logical Block Address**, or **LBA**.
* A drive can only transfer in units of sectors.
* Block sizes vary from **512 bytes** up to **4224 bytes**.

Block Device File Read

- Say we have 512 byte sectors
  - fh = fopen (“myfile”, “r”);
  - buf_ptr = malloc (4000);
  - amount_read = fread(buf_ptr, 1000, 1, fh);



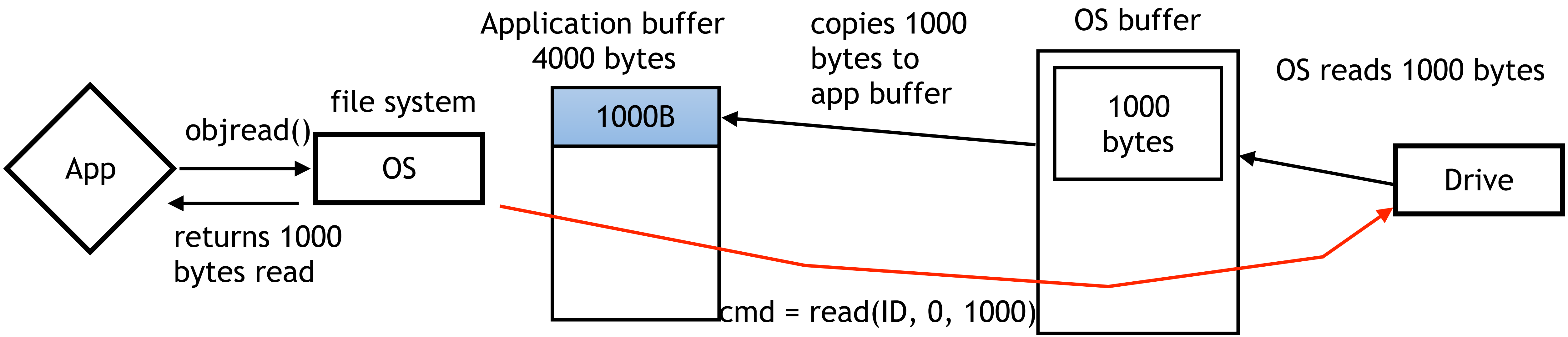
Object Devices



* In object based drives, data is referred to by an **objectID**.
* Objects can be of arbitrary size.

Object Device File Read

- fh = objopen (“myfile”, “r”);
- buf_ptr = malloc (4000);
- amount_read = objread(buf_ptr, 1000, fh);



Key-Value Devices

Host System ⟷ Interface ⟷ Drive

* In key-value based drives, data is referred to by a **key**.
* Values can be of arbitrary size.

time

Read command
key
offset = 0
length = bytes

Read data stream

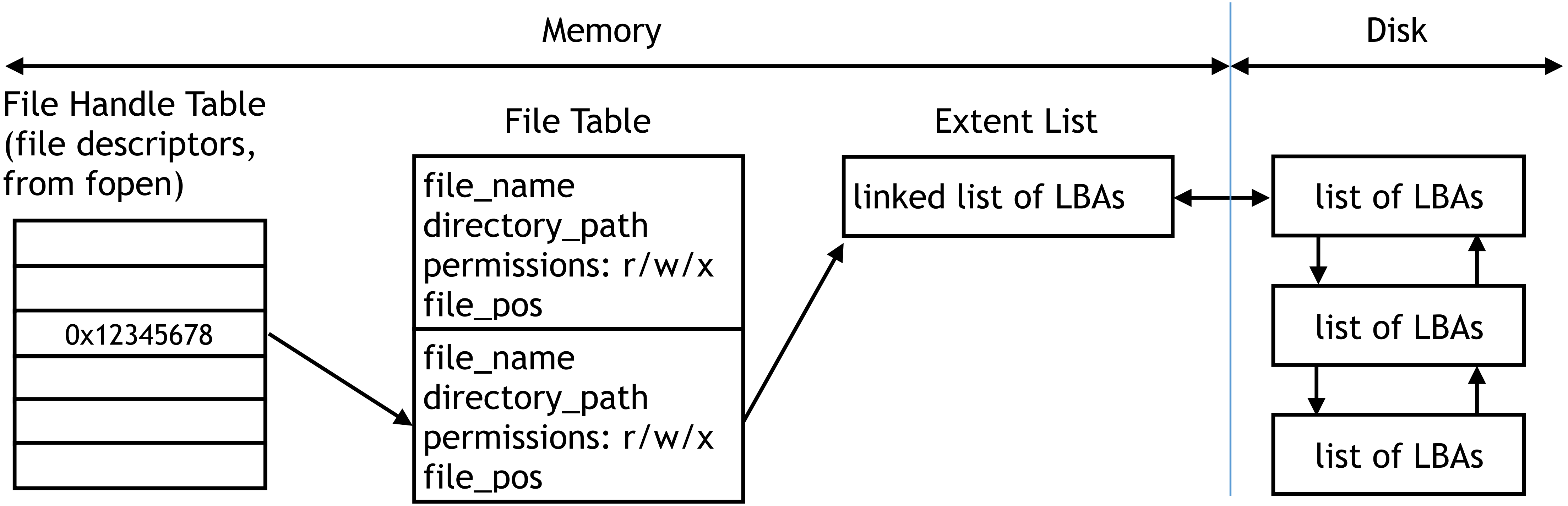
Status

Sources: http://www.enterprisestorageforum.com/storage-management/is-key-value-data-storage-in-your-future-1.html
https://en.wikipedia.org/wiki/NoSQL#Key-value_store
http://www.project-voldemort.com/voldemort/

How Does a File System Track Files?

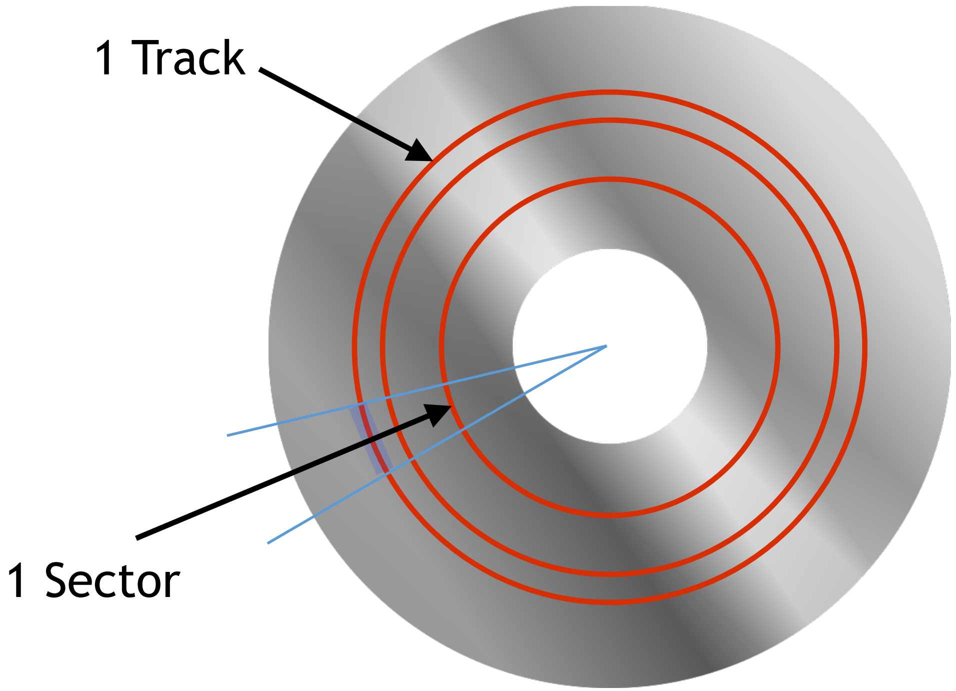
Warning: Cartoon Drawing!

- Disk sectors are used to store:
  - User data (file contents)
  - File system data (directory structure and file metadata)



A File on a Hard Drive

A file on a hard drive is stored in multiple sectors, on multiple tracks, spread around the surface of the disk.



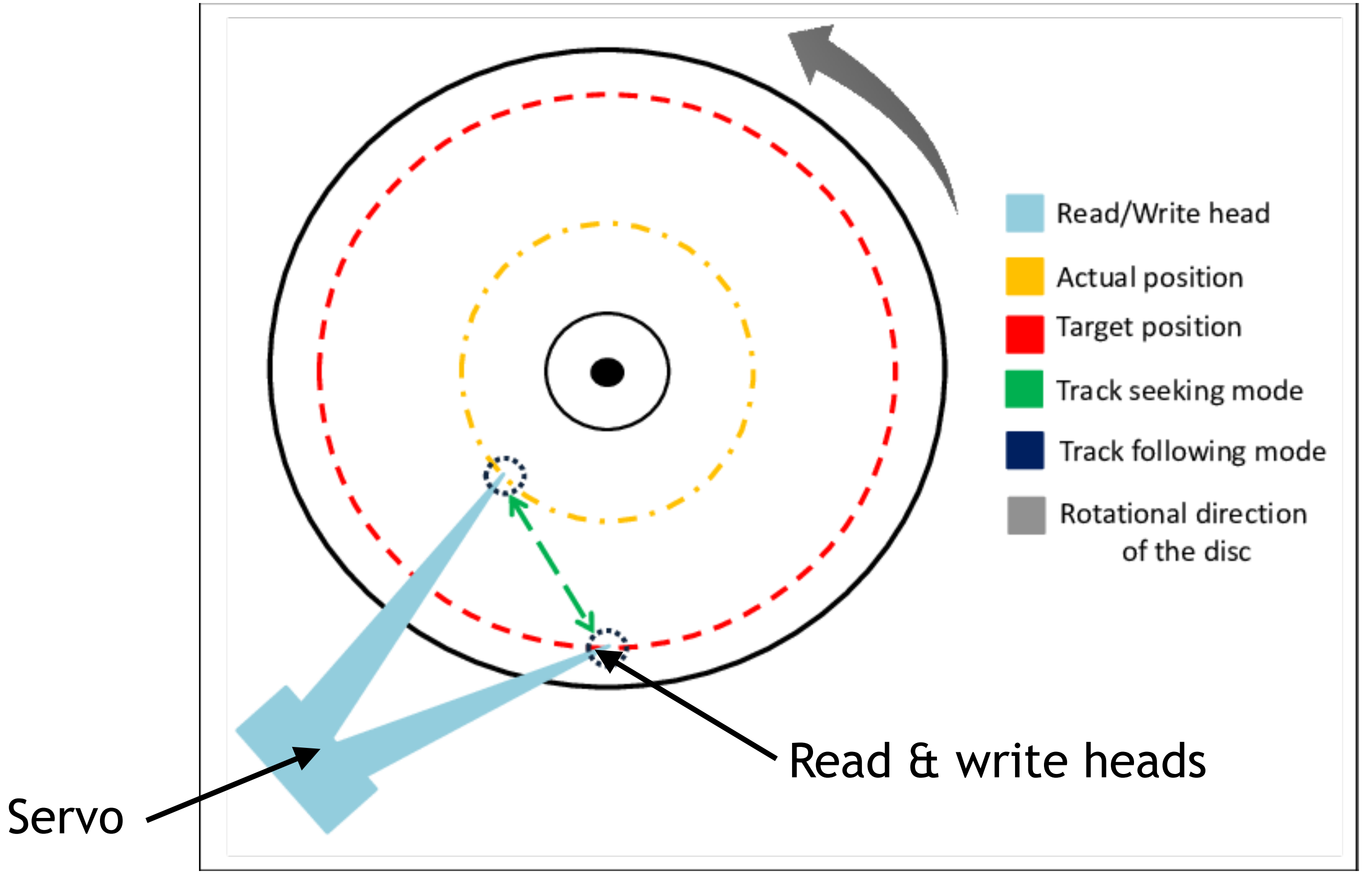
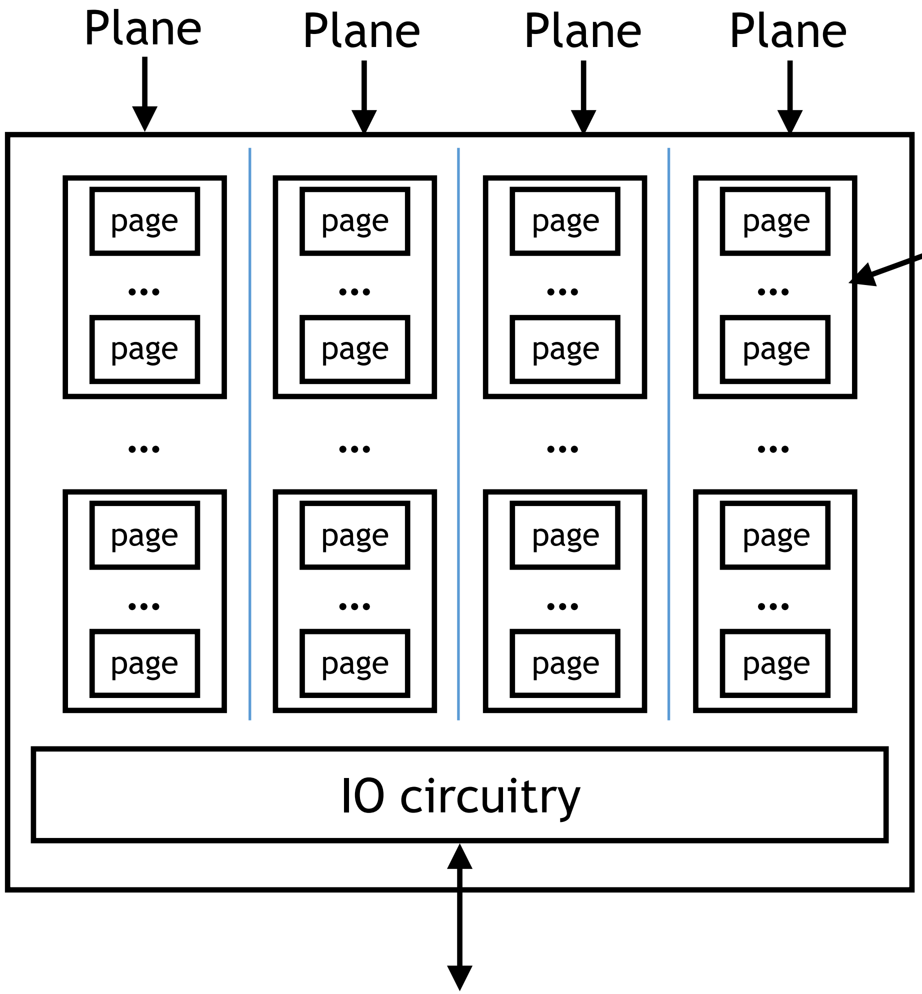


Image Source: https://www.researchgate.net/profile/Jawhar_Ghommam

A File on a Solid State Drive



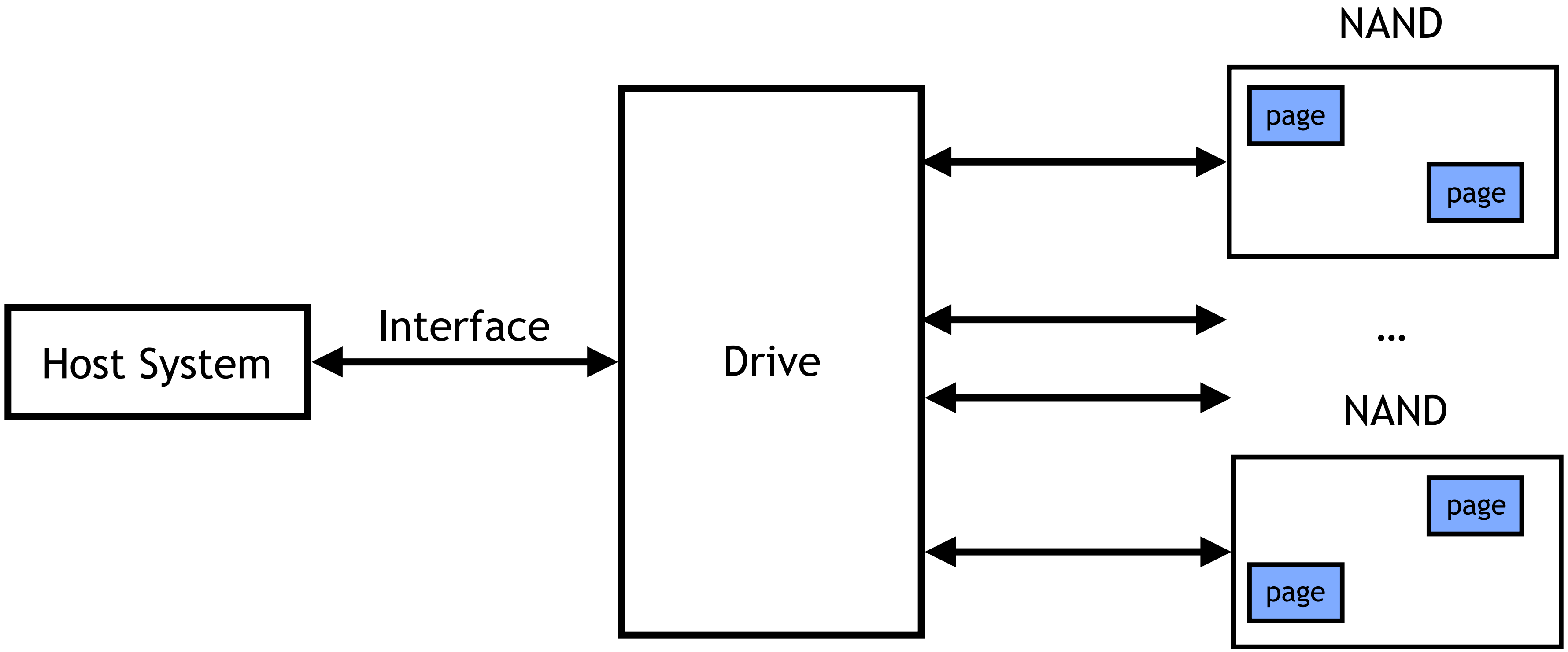
NAND Flash Device

Block

NAND cannot "write-in-place"

- Write pages, once -> have to erase before writing again
- Read pages
- Erase blocks

A File on a Solid State Drive (con't)

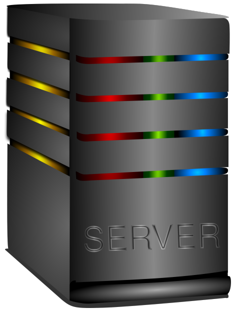


Firmware on the drive maintains a table that maps LBAs to physical locations in the NAND array.

NFS

- NFS = Network File System
- Created by Sun for Solaris
- Client-Server model
    - Allows remote directories to be accessed as though they were local directories
- Runs over TCP/IP
- OS and architecture of the server can be different from the client
    - MS Windows client and a Linux server
    - A folder that contains files resides on a single server (or NAS box)
    - It may be a RAID system for data protection, but generally not for access speed

Windows PC G:\ ⟷ /var/share0/ Linux Server



Drive Speeds

- A hard drive can stream at ~4Gb/s
  - ~ 500 MB/s, limited by the media channel speed
- A Gen4 (1600 MT/s) PCIe/NMVe SSD can stream at 1970 MB/s per lane
  - 4-lane PCIe SSD = ~7 GB/s
  - Assuming the NAND flash array can keep up

So how big is big?

- A manufacturing plant with 30,000 sensors, sampling at 100ms producing 16 bytes of data/sample yields:
  - 30,000 sensors * 1 sample/sensor/100ms * 16 bytes/sample
  - = 48,000 bytes/sec or 4.1 GB/day, 1.5 TB/year
- Multi-wave Infrared Atlas of the Galactic plane, 45 TB
- Pratt and Whitney's Geared Turbo Fan engine has 5000 sensors, generates up to 10GB of data per second. 2 hour flight generates 72 TB
- One Watson typically has 16TB of RAM

Pratt and Whitney source: http://aviationweek.com/connected-aerospace/internet-aircraft-things-industry-set-be-transformed

Sizes

- Terabyte (TB) = 1000 GB
- Petabyte (PB) = 1000 TB
- Exabyte (EB) = 1000 PB
    - Estimate today (Q1 2018), the internet contains ~170 EB
- Zetabyte (ZB) = 1000 EB

Bottlenecks of Traditional File Systems

100G ethernet

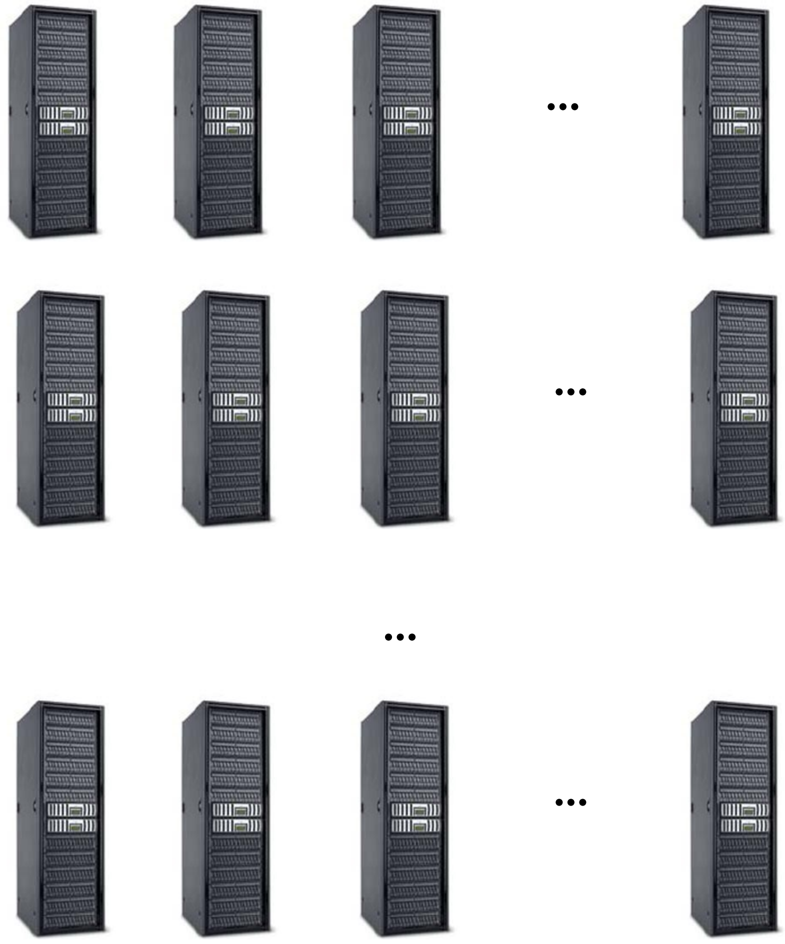
Transfer 2880 TB / 11.25E9 B/sec
= 256,000 sec
= 4267 minutes
= 71 hours

NetApp FAS3200
2880 TB (2.88 PB)

Bottlenecks (con't)

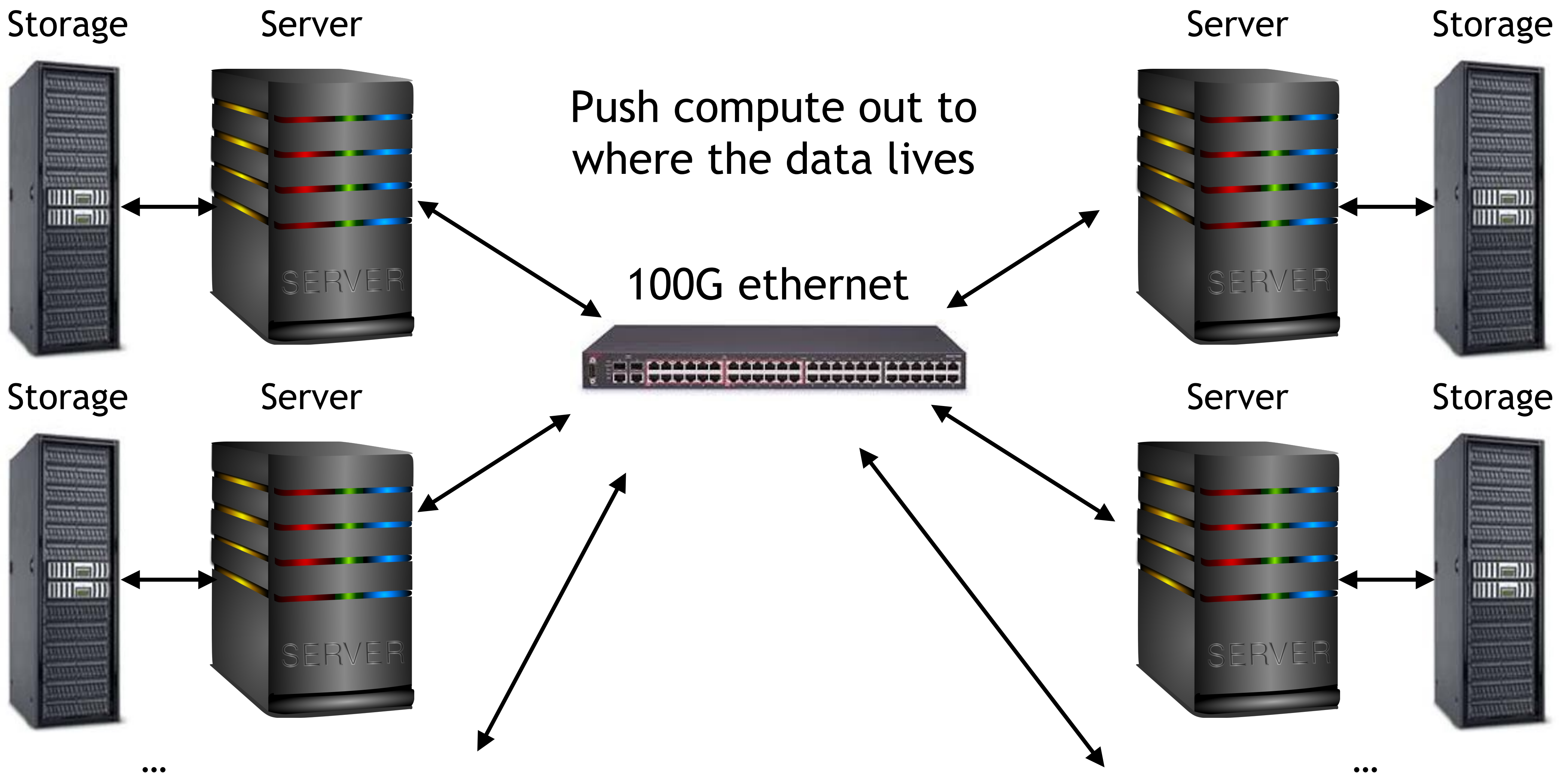
128 NetApp FAS3200's

100G ethernet



Uh oh. The pipe may be too narrow!

Better Scheme



Parallel Distributed File Systems

- Hadoop File System (HDFS)
- Lustre (LFS)

Note: This is a very similar notion to "Storage Compute", which also aims to push compute out to where data resides.
Primary driver in Storage Compute is:

1) Throughput, parallel access to data
2) Dramatically reduced power consumption, reduce the power cost for moving the data to the compute

Hadoop

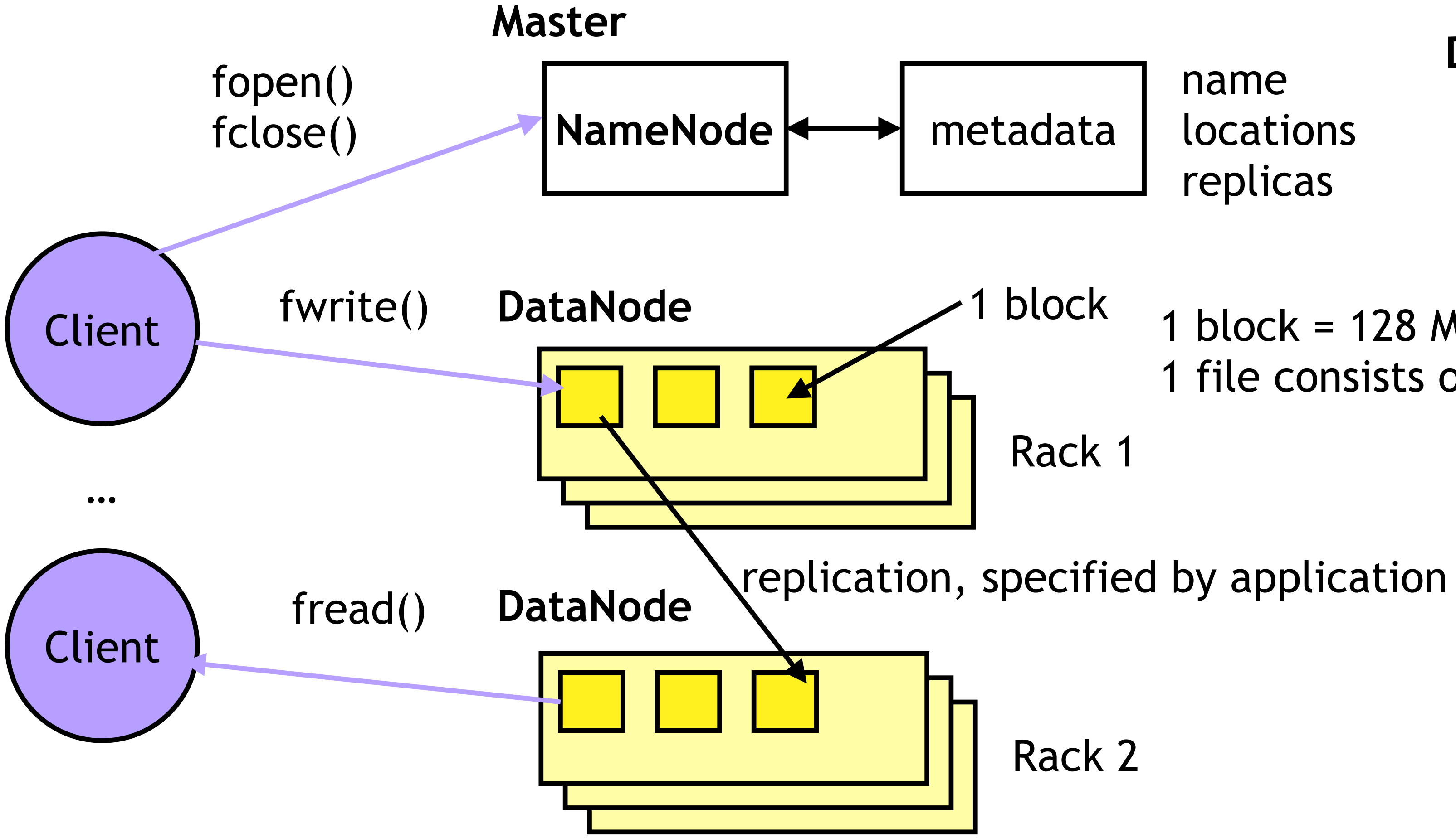
- Goals
  - Plan for and mitigate hardware failures
  - Streaming access to data and batch processing
  - Large data files, a typical file in Hadoop is GB's to TB's
  - Is a **write-once, read-many** scheme. Does support append and truncate. Not a good fit for general purpose file IO.
  - Supported and portable across heterogeneous hardware and software systems.

Source: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Organization

Example Hadoop cluster

NameNodes perform file:
- open
- close
- renaming

DataNodes perform block:
- creation
- deletion
- replication



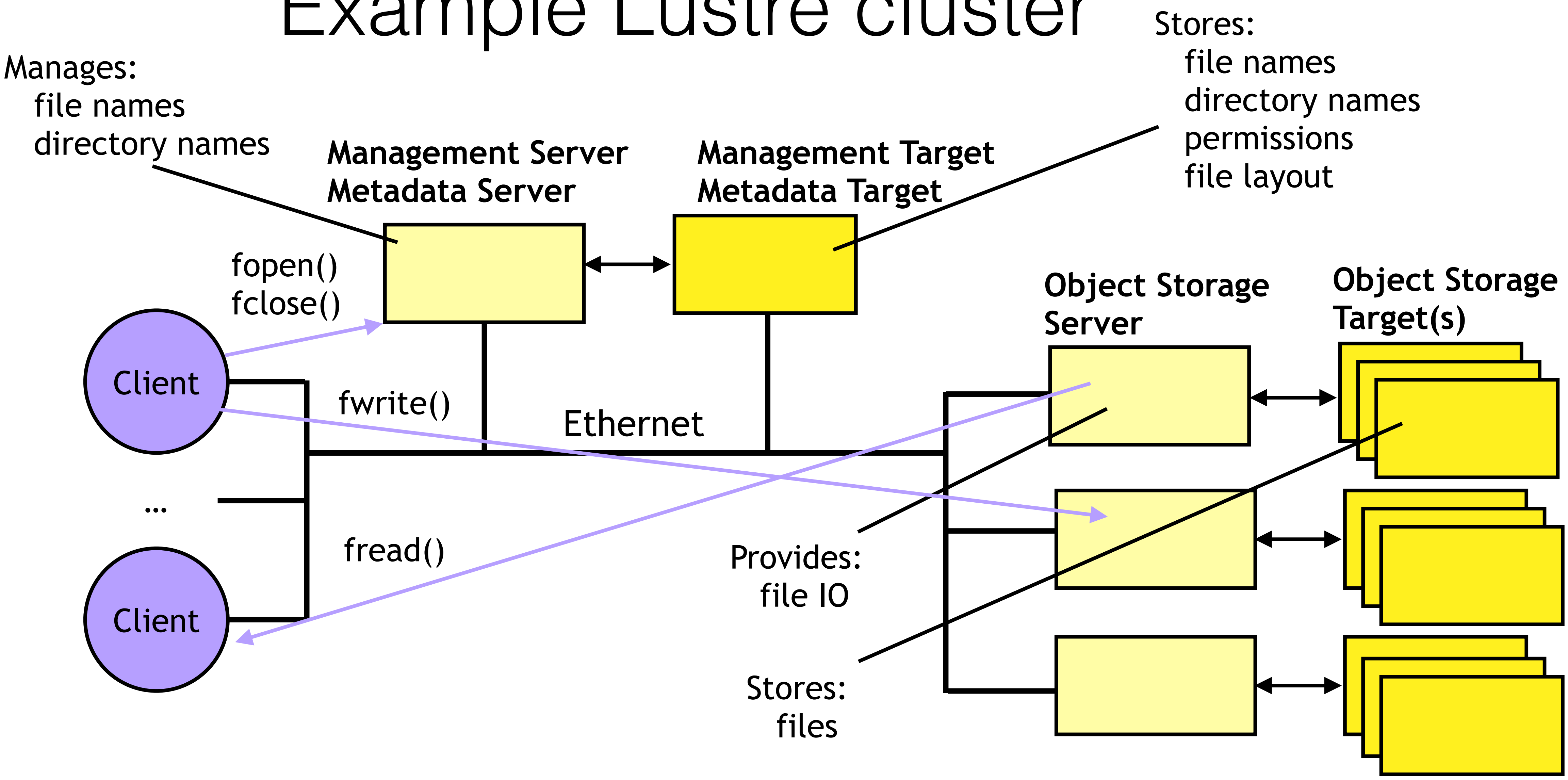
1 block = 128 MB, typ, but configurable
1 file consists of multiple blocks

Lustre

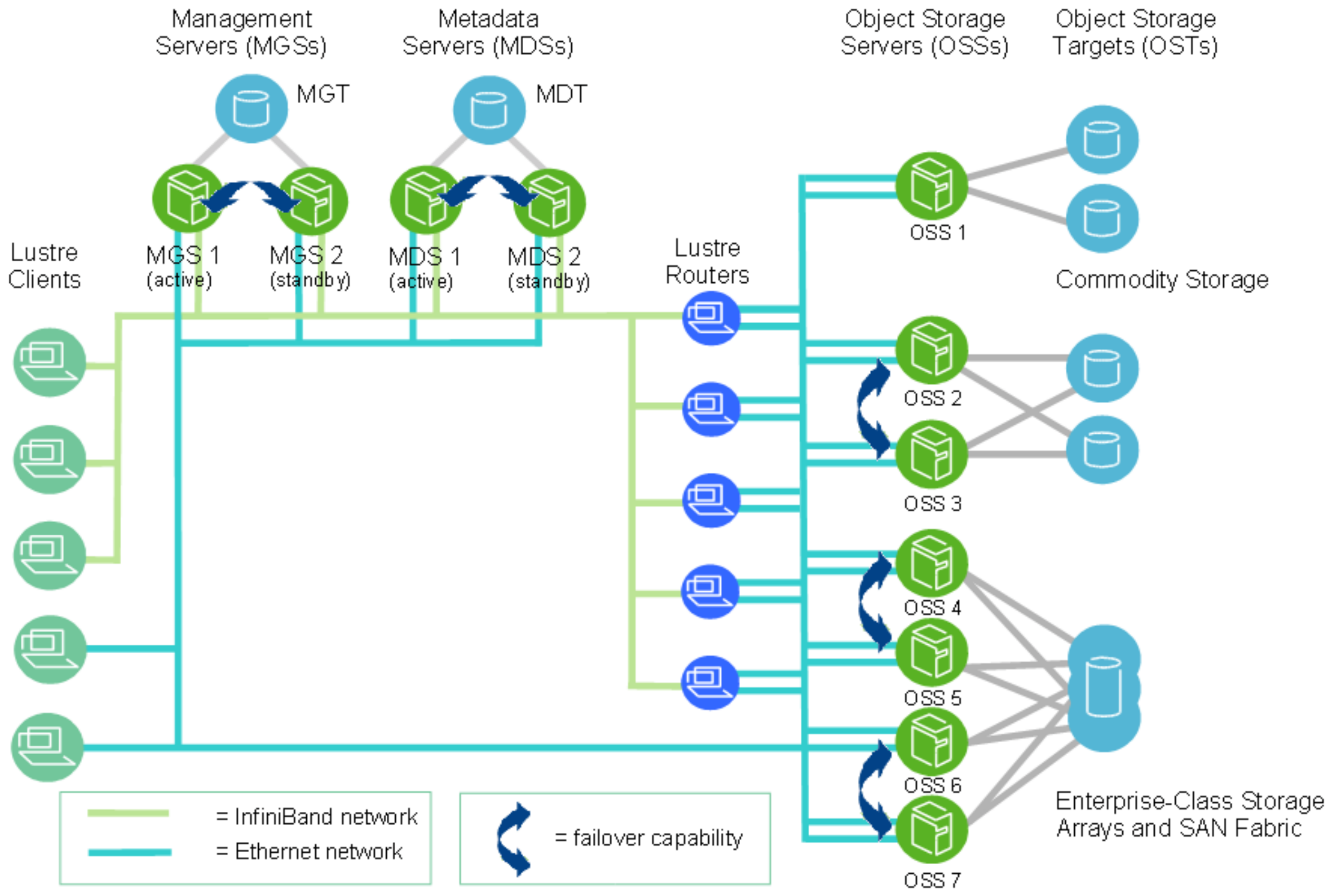
- Goals
  - Run on linux
  - Scale capacity and performance
  - Number of Clients = 100 to 1,000,000; Installations today with 50,000+ clients
  - Client performance
    - Single client = 90% of network bandwidth
    - Aggregate client bandwidth can reach up to 10 TB/s
  - File sizes
    - 32 PB max file size
    - 512 PB, with over 1 trillion files
  - General file IO, not write-once read-many like Hadoop
  - Replication not a focus, but can support

Source: http://doc.lustre.org/lustre_manual.xhtml#understandinglustre
http://wiki.lustre.org/File_Level_Replication_High_Level_Design

Example Lustre cluster



Lustre at scale



End