DS5001 Advanced Big Data Analytics

sp2019, sp20, sp21, sp22, sp23, **sp 2024** (6)

Omar Usman Khan
omar.khan@nu.edu.pk

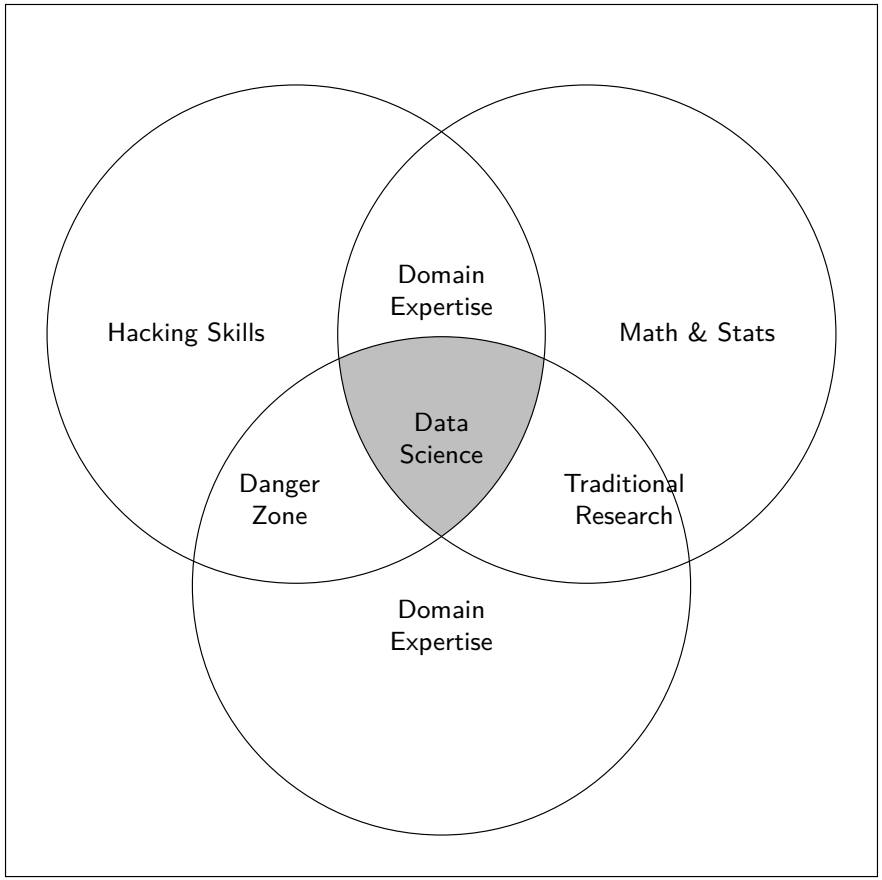
Department of Computer Science
National University of Computer & Emerging Sciences, Peshawar

May 8, 2025

Syllabus

1. Overview
   - Industry
   - This Course
   - Data / Hardware / System
   - Speedup
   - Data Analytics Pipeline
2. Hadoop
   - HDFS
   - HDFS API's
   - Map Reduce Framework
3. Spark
4. CUDA
   - Overview
   - GPGPU's: OpenCL & CUDA
   - CUDA Programming
   - Optimization
5. Network Programming
   - Socket Programming
   - ZeroMQ
6. Kafka
7. Spark ML & MLLib

Big Data in Context of Data Science



W. Cleveland, *Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics*, in InternationaL Statistical Review, 69(1), 2001

Overview

Understanding the Scale

- New York Stock Exchange: 1 TB / Day
- Large Hadron Collider: 15 PB / Year
- Internet Archive: 2 PB stored, Growing @ 20 TB / month
- Surveillance Drones (Multiple cameras, several TB / minute)
- Facebook, Google, Twitter, Instagram, Youtube, ...
- ...

- Amount of Data Overwhelms Analysts & Decision Makers

Characteristics

Collections of datasets whose **Volume**, **Velocity**, and **Variety** is so large that it is difficult to **Store**, **Manage**, **Process**, and **Analyze** the data using traditional databases or processing tools.

Job Prospects

Job Title	Salary (Entry)	Salary (Medium)	Salary (Senior)
(Big) Data Engineer, Business Analyst, Data Analyst	74,110	118,115	144,770
Data Scientist	64,674	119,478	233,470

Table 1: P@sha IT Salary Survey 2017

Sample Entry Requirements

- Technology expertise of solutioning in Hadoop, Hive, Spark / PySpark, SQL, Oozie along Data Modelling in Hive
- Expertise in programming Languages- Java / Python / Scala
- Ability to demonstrate micro / macro designing and familiar with Unix Commands and basic work experience in Unix Shell Scripting
- ...

Job Prospects (cont.)

Companies

- IBM
- Afiniti
- Telecom (Mobilink, Telenor, Zong)
- Banks (Allied)
- Logistics (Careem, Bykea, Keep Trucking)
- 10Pearls, i2c
- ...

Job Prospects (cont.)

Number of Jobs

Job Role	Pakistan	Middle East	Europe	USA
Data Scientist	7	197	6011	6561
Data Analyst	15	160	4845	8564
Data Manager	0	11	1283	1255
Data Architect	4	27	1220	2184
Business Analyst	9	212	9595	15244
Financial Analyst	2	69	1194	6395
Big Data Engineer	3	24	603	674
Data Engineer	6	135	5894	6838
Machine Learning Specialist	1	3	46	14

Table 2: DS Jobs Worldwide (Source: LinkedIn, fetched October 13, 2020)

Job Prospects (cont.)

Resources Required Programming and Data Analyst

A total of **3146 resources** are required in the programming and data analyst domain with the highest demand for **Code Repository (SVN, Git, etc...)**

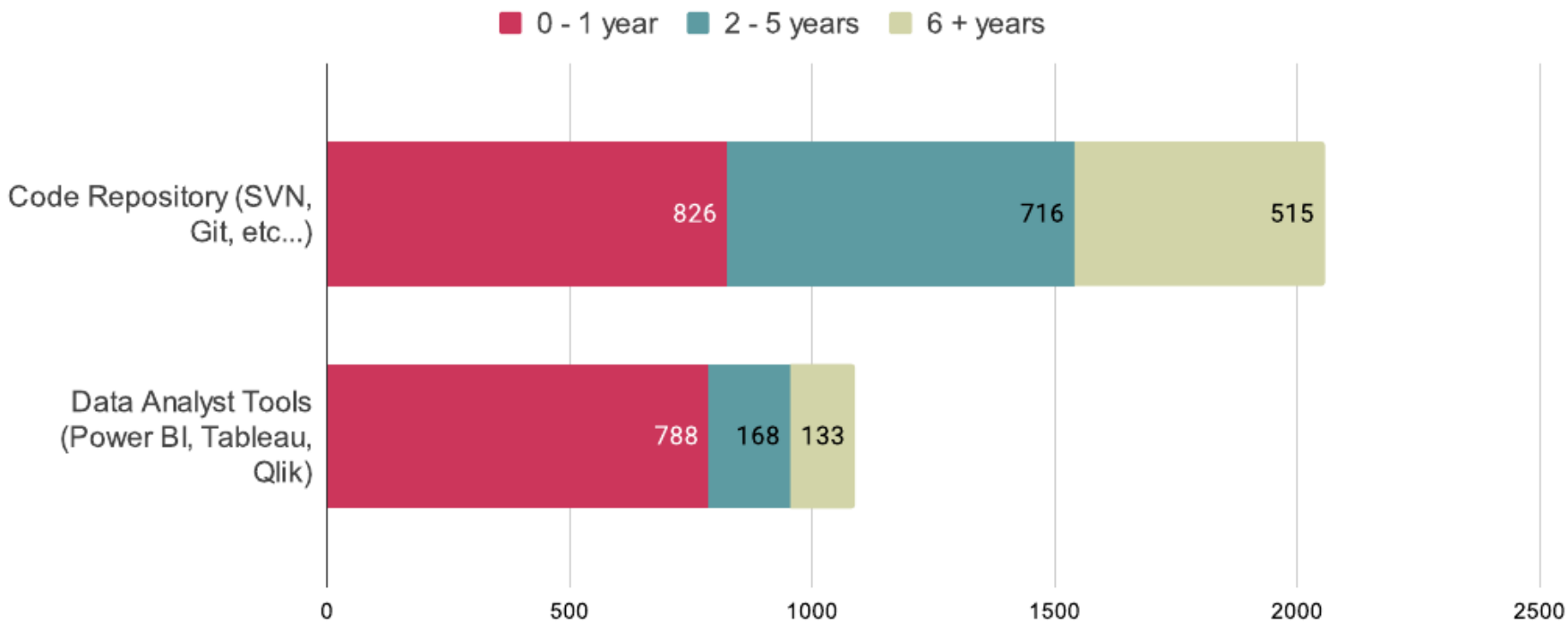


Figure 1: P@sha Skill Report November 2023

Job Prospects (cont.)

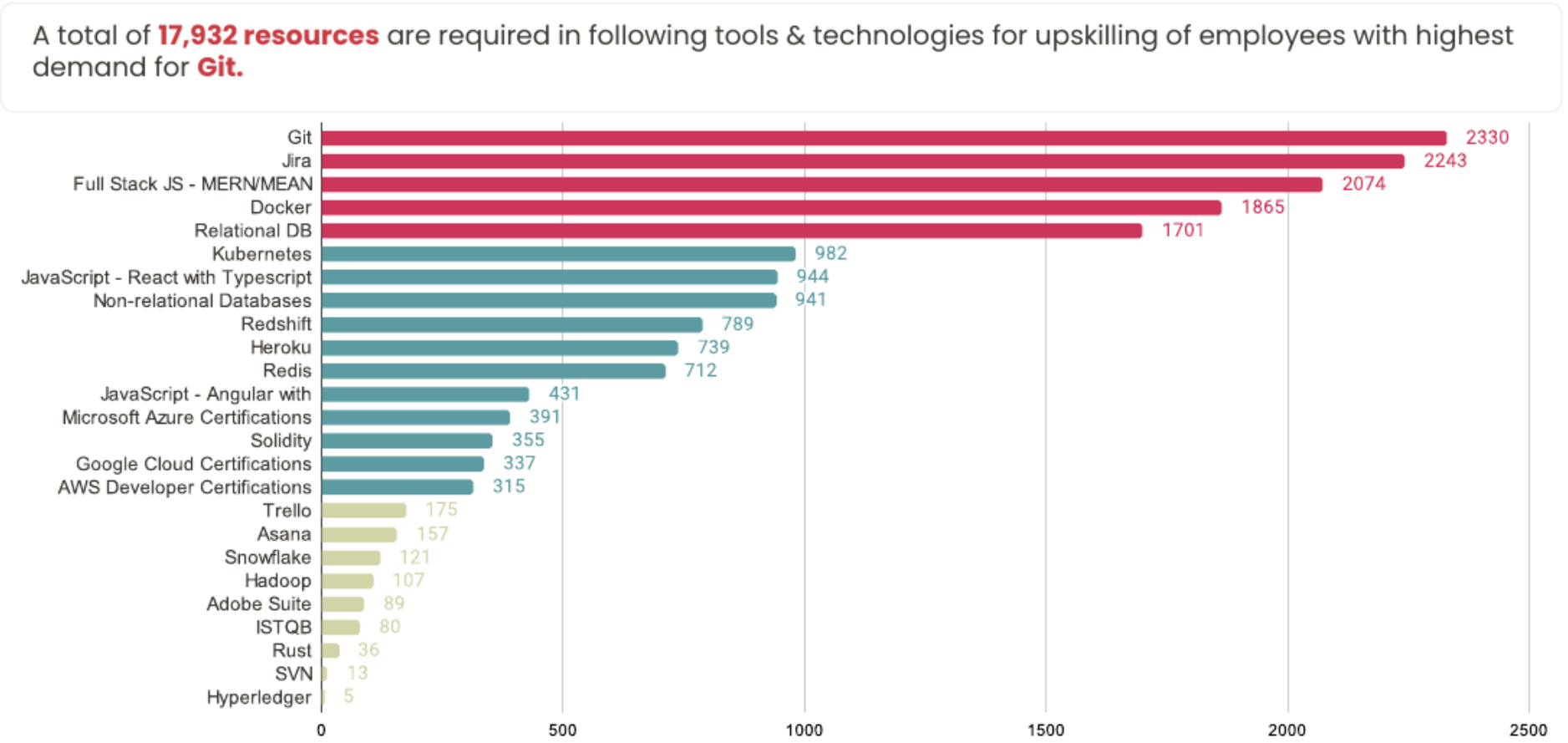


Figure 2: P@sha Upskill Report November 2023

Job Prospects (cont.)

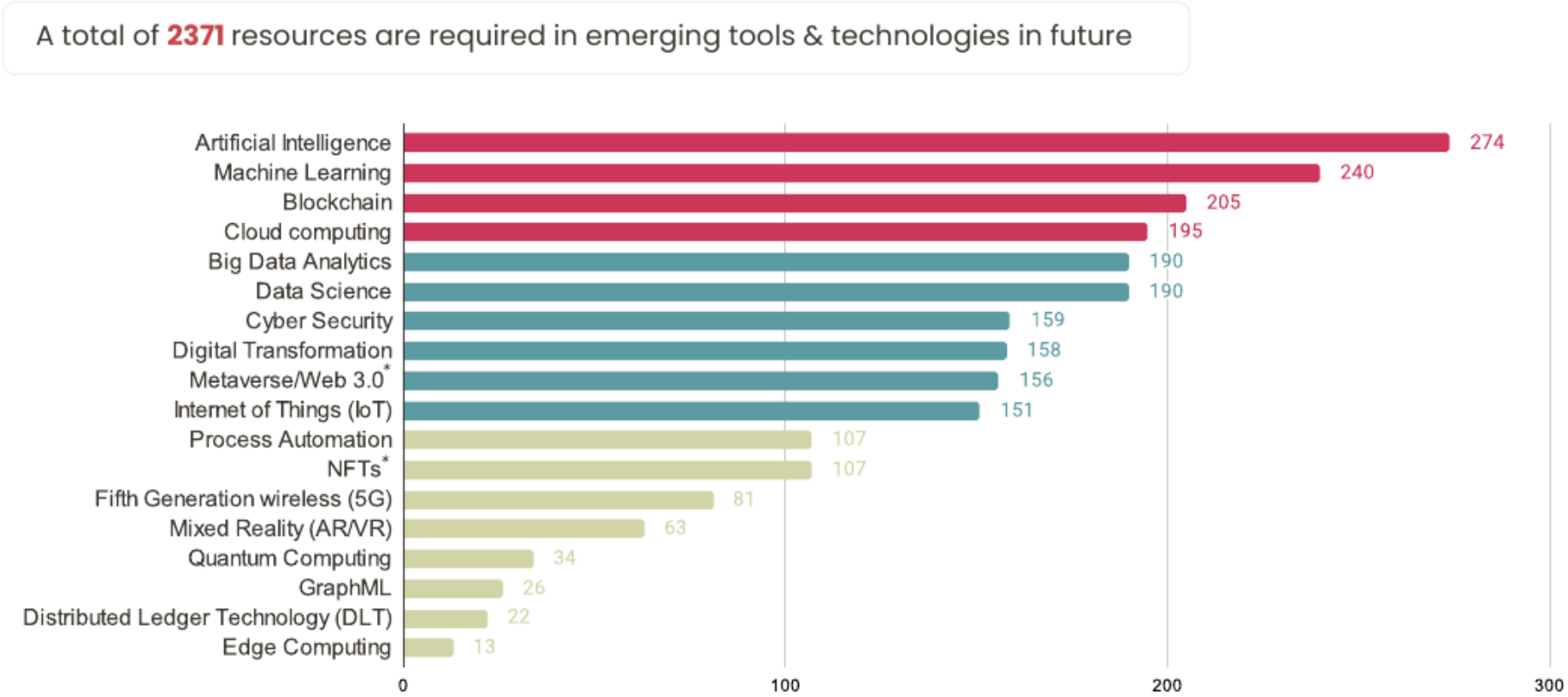


Figure 3: P@sha Emerging Skill Report November 2023

Job Prospects (cont.)

Figure 4: On a Lighter Note

Course Overview

- Classroom Code: fyvt7ff

Learning Outcomes

CLO1	Perform analytics on large scale datasets
CLO2	Deploy massive threading solutions on Parallel systems
CLO3	Deploy massive threading solutions on Distributed systems
CLO4	Be able to visualize Large Scale Multi-Dimensional Datasets
CLO5	Be familiar with algorithms and tools for mining massive datasets

Marks Breakdown

- Sessional I: 15%
- Sessional II: 15%
- Final Examination: 50%
- Assignments: 20% (including Takehome Lab Activities)

Course Overview (cont.)

- HPC Setup with support for various parallel and distributed computing frameworks will be made available to all students for duration of semester (and beyond upon request)

Generating Public Key for Assignments

- On Windows, Use https://winscp.net/eng/docs/ui_puttygen
- On Linux, use **ssh-keygen** command
- Share generated key with me by email.

Course Overview (cont.)

Probability Distribution of Previous Grades

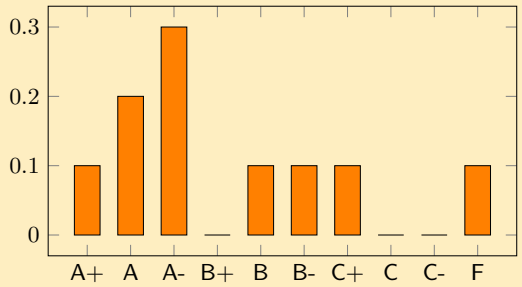


Figure 5: Spring 2019

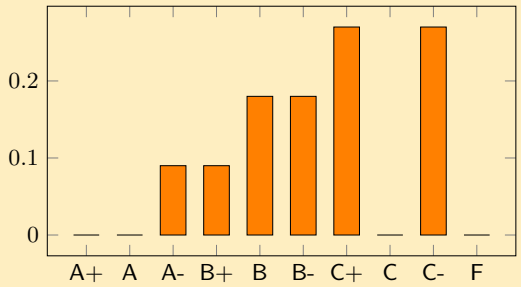


Figure 6: Spring 2020

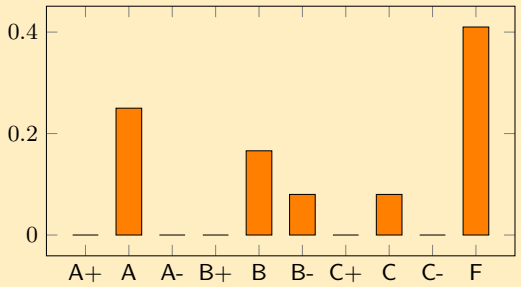


Figure 7: Spring 2021

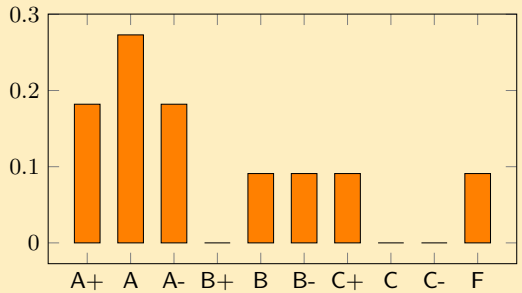


Figure 8: Spring 2022

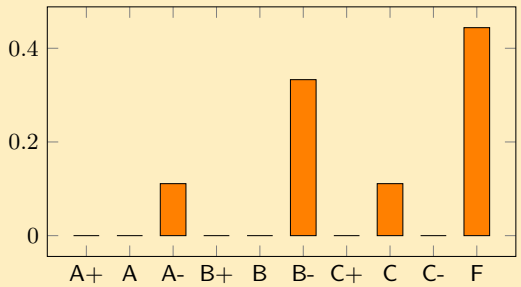


Figure 9: Spring 2023

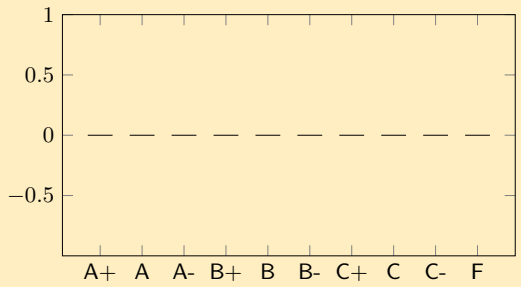
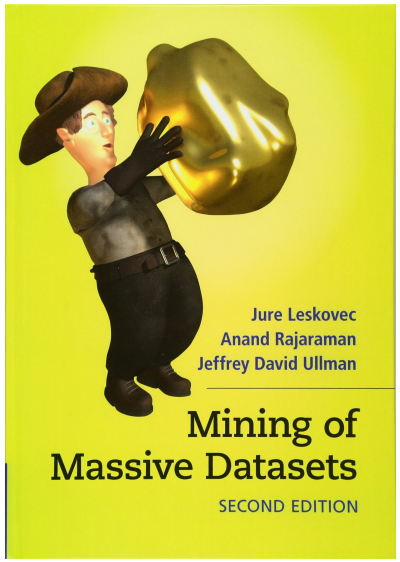
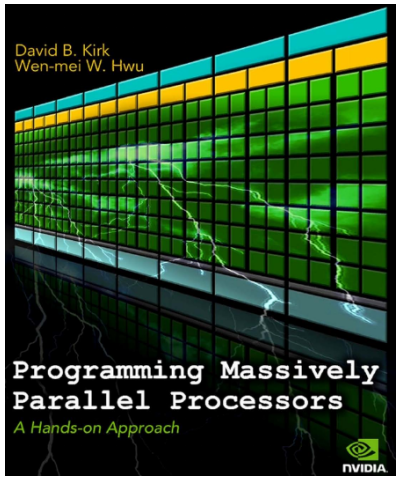
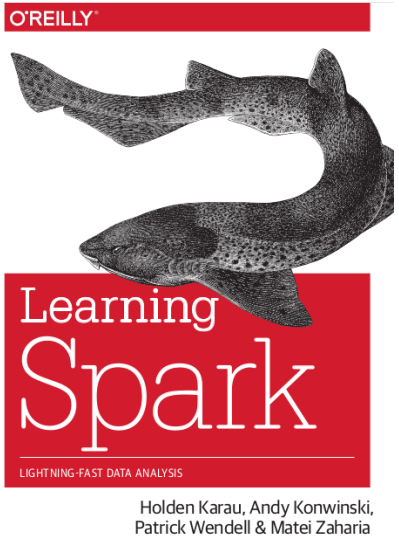


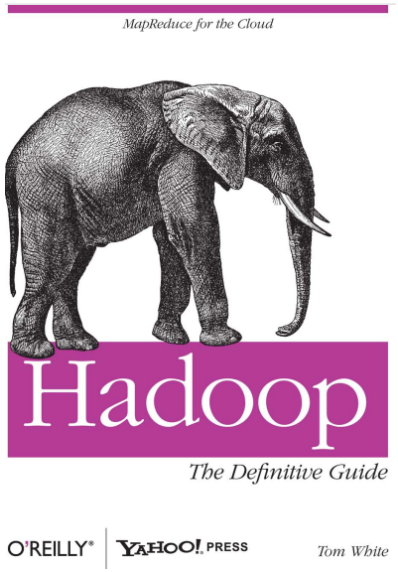
Figure 10: Spring 2024

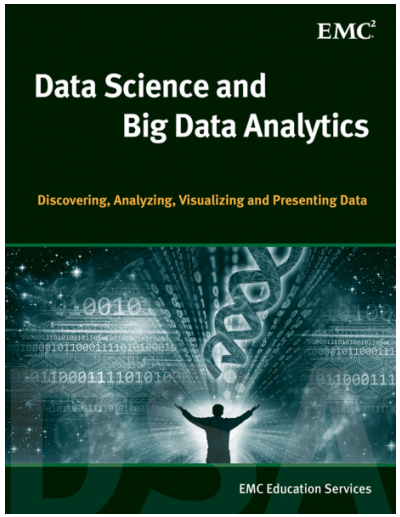
Books


Jure Leskovec
Anand Rajaraman
Jeffrey David Ullman
Mining of Massive Datasets
SECOND EDITION


David B. Kirk
Wen-mei W. Hwu
Programming Massively Parallel Processors
A Hands-on Approach
NVIDIA


O'REILLY
Learning Spark
LIGHTNING-FAST DATA ANALYSIS
Holden Karau, Andy Konwinski,
Patrick Wendell & Matei Zaharia


MapReduce for the Cloud
Hadoop
The Definitive Guide
O'REILLY
YAHOO! PRESS
Tom White


EMC²
Data Science and Big Data Analytics
Discovering, Analyzing, Visualizing and Presenting Data
EMC Education Services

Data

Raw Data Sources

- **Logs**: Web applications/servers/daemons
- **Transactions**: E-Commerce/Banking/Financial
- **Social Media**: JSon
- **Databases**: RDBMS
- **Sensor Data**: IoTs/WSNs
- **Clickstream Data**: Patterns of user activities
- **Surveillance**: Sensors/Images/Video Data
- **Healthcare**: Sensors/Hospital Records
- **Network**: Info generated by Network Devices
- ...

Data Storage

- Issues: Disk Density, Access Times, Storage Formats
- Parallel Disk Access & Distributed Storage Solutions

Data (cont.)

Data Processing

- Combining Data (Parallel Access, Distributed Storage)
- HPC and Super-Computing Systems
- MPI, GPU Computing, ...

Think!

Which code is going to be the fastest?

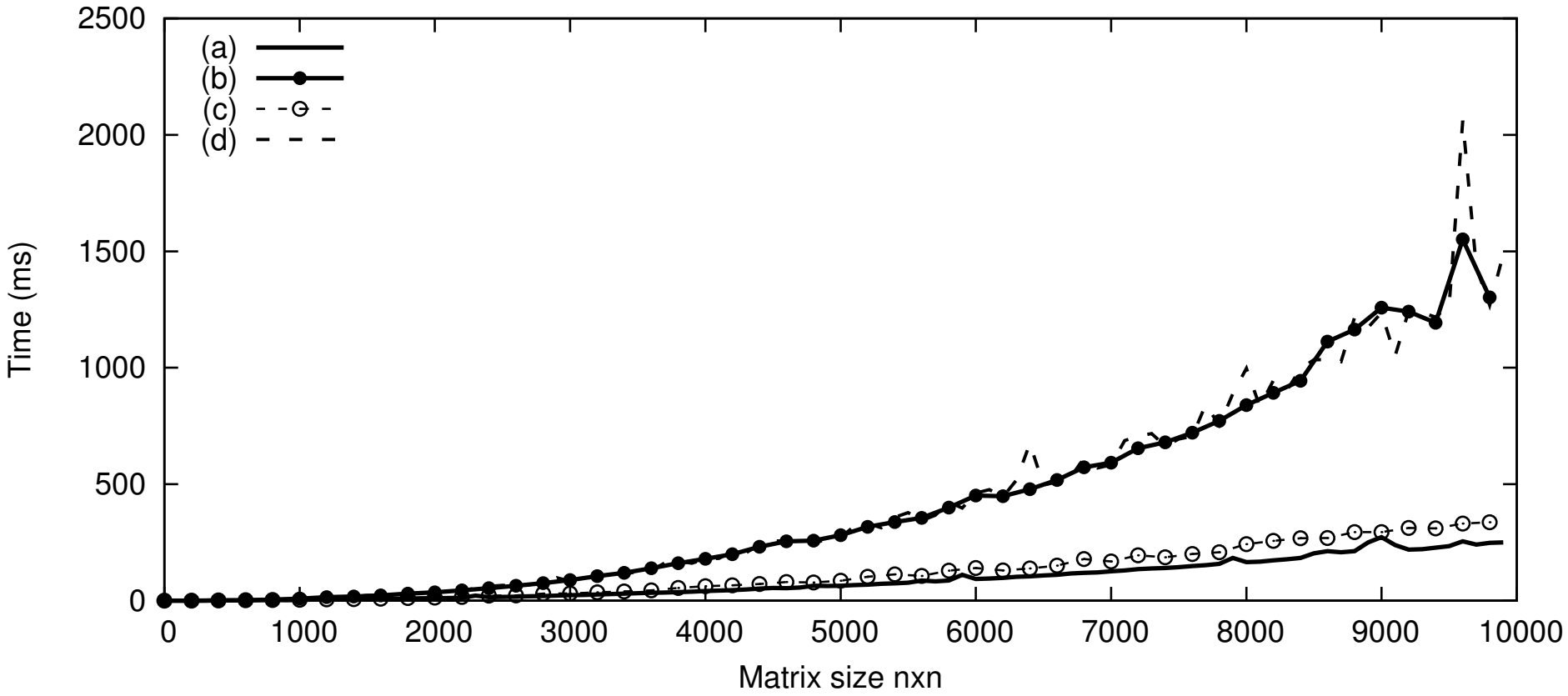
(a)
```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    sum += a[i][j];
```

(b)
```
for (j = 0; j < n; j++)
  for (i = 0; i < n; i++)
    sum += a[i][j]
```

(c)
```
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    sum += a[i*SIZE+j];
```

(d)
```
for (j = 0; j < n; j++)
  for (i = 0; i < n; i++)
    sum += a[i*SIZE+j]
```

Think! (cont.)


(a)
(b)
(c)
(d)
Time (ms)
Matrix size nxn
0
500
1000
1500
2000
2500
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Clock Cycles

- What is a single clock cycle worth?
  - **xchg**: Exchange values in two registers
  - **push**: Push operation using registers
  - **pop**: Pop operation using registers
  - **add**,**sub**: 3 additions/subtractions involving registers
  - **cmp**: 3 comparisons involving registers
  - **mul**: half a fp multiplication involving registers

Where are We Today?

- Clock rates 40 MHz (MIPS R3000 1988) $\rightarrow$ 4.0 GHz (Intel Core-i7-4790K 2015), 4.4 GHz (Intel Xeon X5698 2015)
- Higher processor speed $\propto$ More **heat dissipated**
- Transistor has reached size of 32 nm (Generation 3 Core-i7). **Size limit**: How much more smaller can it get?
- Support for executing multiple instructions per clock cycle, fast cache technologies, superscalar architectures ...

Ranking of Super Computers

Rank	Site	System	Cores	TFLOPs	Power (kWh)
1	Wuxi, China	TaihuLight: Sunway 1.45Ghz	10,649,600	125,435.9	15,371
2	Guangzhou, China	Tianhe-2: Intel Xeon	3,120,000	54,902	17,808
3	Oak Ridge Lab, USA	Titan: Cray Opteron, NVIDIA K20	560,640	27,112	8,209
4	DoE, USA	Sequoaia:, IBM BlueGene/Q	1,572,864	20,132	7,890
5	RIKEN Ins., Japan	K-Computer: Fujitsu SPARC64	705,024	11,280	12,660
6	DoE, USA	Mira: IBM BlueGene/Q	786,432	8,586.6	3,945
7	DoE, USA	Trinity: Cray XC40	301,056	11,078	-
8	Swiss S.Comp., Switzerland	Cray, Intel Xeon, NVIDIA K20	115,984	7,788	2,325
9	HLRS, Stuttgart, Germany	Hazel Hen: Cray XC40, Intel Xeon	185,088	7,403.5	-
10	KAUST, S. Arabia	Shaheen: Cray XC40, Intel Xeon	196,608	7,235.2	2,834
-	Your Home	Intel Core-i7	4	.026	0.3
-	Your Home	NVIDIA K40	2,880	4.29	0.3
-	Your Home	NVIDIA K80	4,992	8.73	0.3

Table 3: Top 10 Supercomputers: June 2016, top500.org

Ranking of Super Computers (cont.)

Rank	Site	System	Cores	TFLOPs	Power (kWh)
1	RIKEN, Japan	Fugaku: Fujitsu 2.2 GHz	7,630,848	442,010	29,899
2	Oak Ridge Lab, USA	Summit: IBM 3GHz, NVIDIA Volta GV100	2,414,592	148,600	10,096
3	DoE, USA	Sierra: IBM 3.1 GHz NVIDIA Volta GV100	1,572,480	94,640	7,438
4	Wuxi, China,	TaihuLight: Sunway 1.45GHz	10,649,600	125,435	15,371
5	DoE, USA	PerlMutter: Cray 2.4GHz, NVIDIA A100	761,856	70,870	2,589

Table 4: Top 10 Supercomputers: Nov 2021, top500.org

Ranking of Super Computers (cont.)

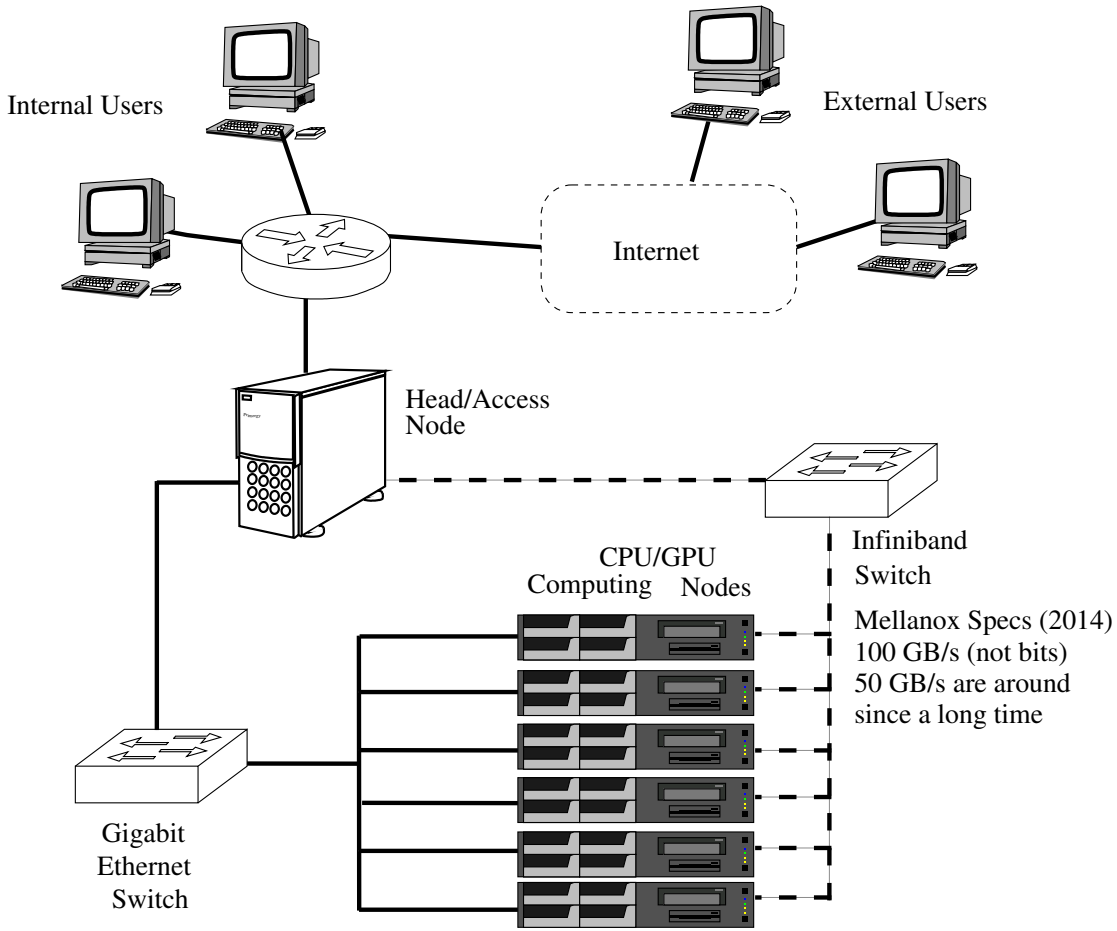


Figure 11: Setup of a typical HPC facility

Moore's Law, 1965

A prediction that would define the pace of digital revolution. Many interpretations:

- Computing would increase in **power** exponentially
- Computing would decrease in relative **cost** exponentially
- **Transistory density** will double every year (revised to double every 18 months)

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

By Gordon E. Moore

Director, Research and Development Laboratories, Fairchild division of Fairchild Camera and Instrument Corp.

The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

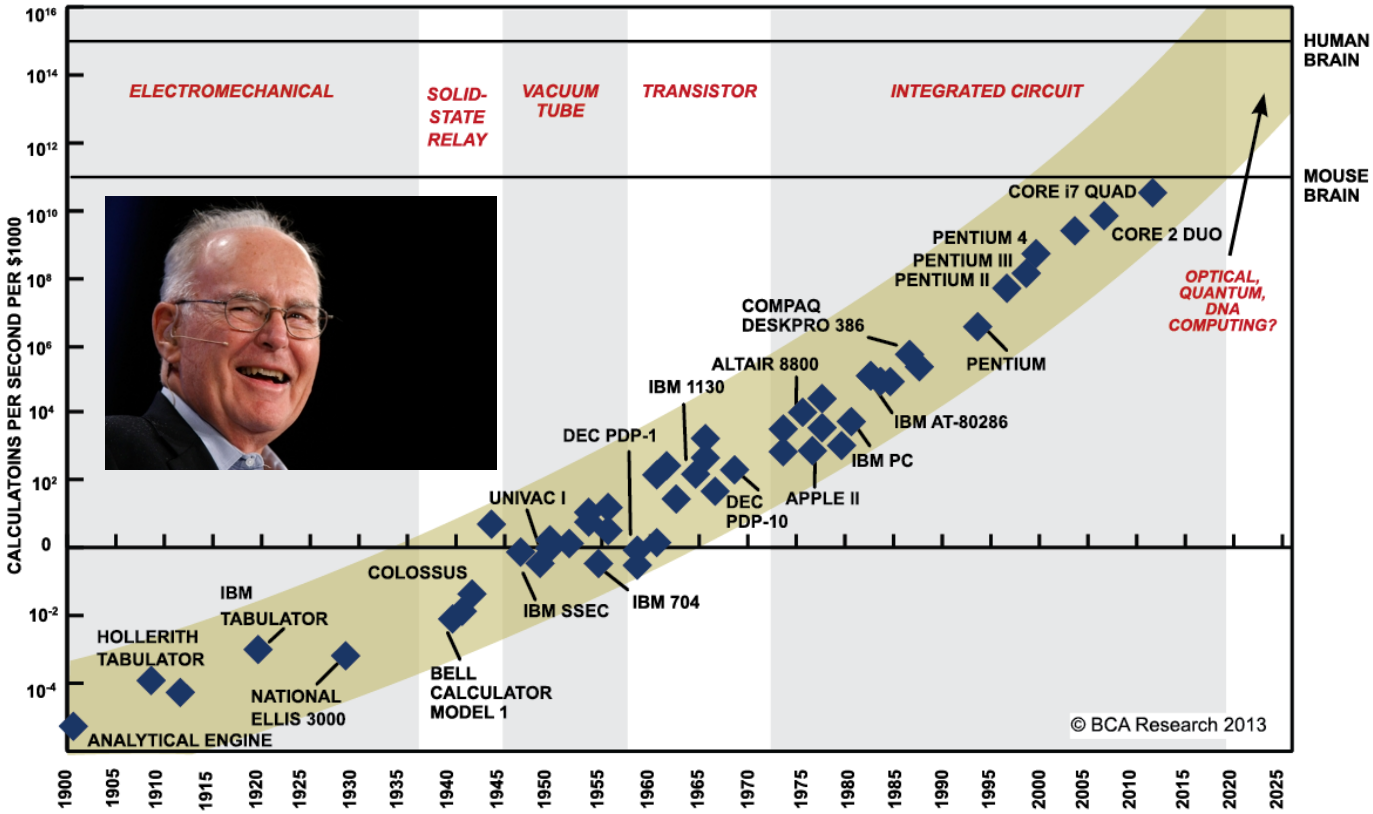
But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the

The author

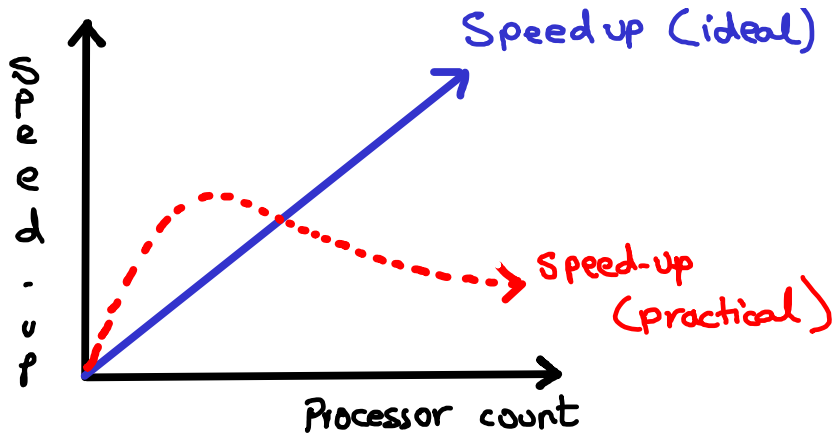
Dr. Gordon E. Moore is one of the new breed of electronic engineers, schooled in the physical sciences rather than in electronics. He earned a B.S. degree in chemistry from the University of California and a Ph.D. degree in physical chemistry from the California Institute of Technology. He was one of the founders of Fairchild Semiconductor and has been director of the research and development laboratories since 1959.

Electronics, Volume 38, Number 8, April 19, 1965



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Premise: Can software exploit this hardware for speed??



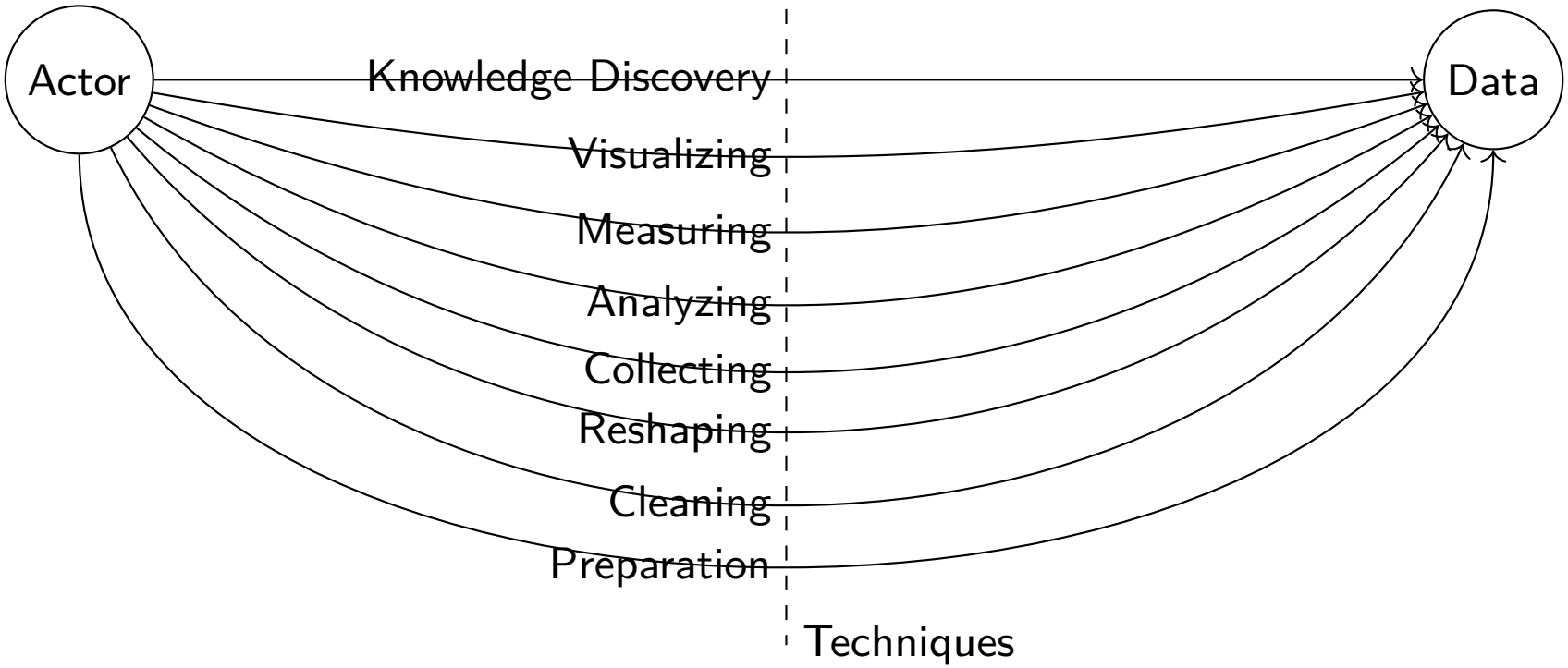
- Why ideal speedup is not possible: data transfer (through message exchanges), I/O bottlenecks, race conditions, dependencies, contention (critical section), load balancing, deadlocks, synchronization, node failures, ...
- **Programmers lacking skills in parallel & distributed regime** ...

Amdahls Law

$$S = \frac{1}{(1-P) + \frac{P}{N}} \tag{1}$$

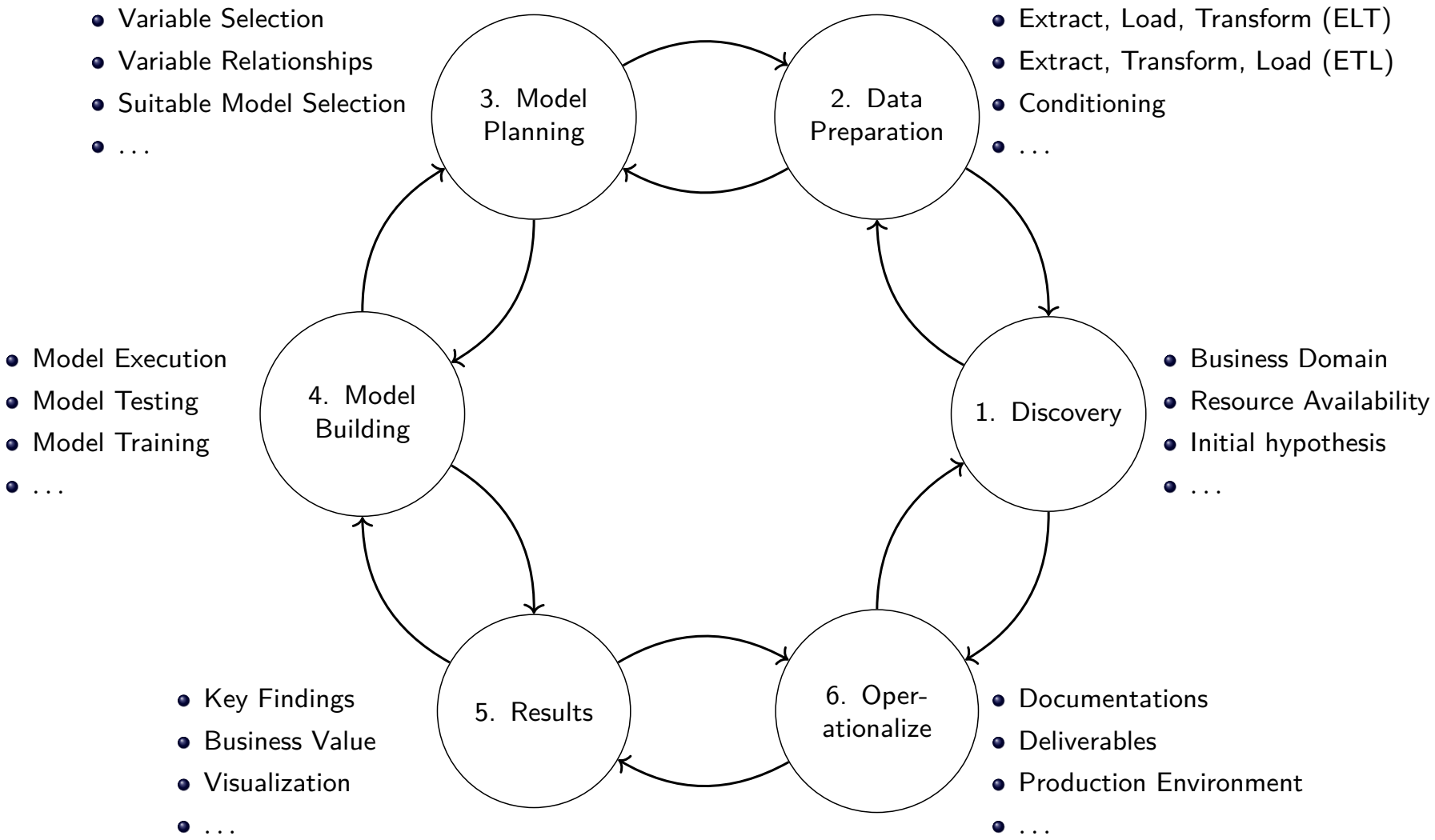
P: Portion of Parallel Region of Code
1-P: Portion of Serial Region of Code
N: Number of Processors

Analytics Pipeline



- Data scale may render techniques useless
- New techniques (from other discplines) employed to guarantee functioning of operations on data

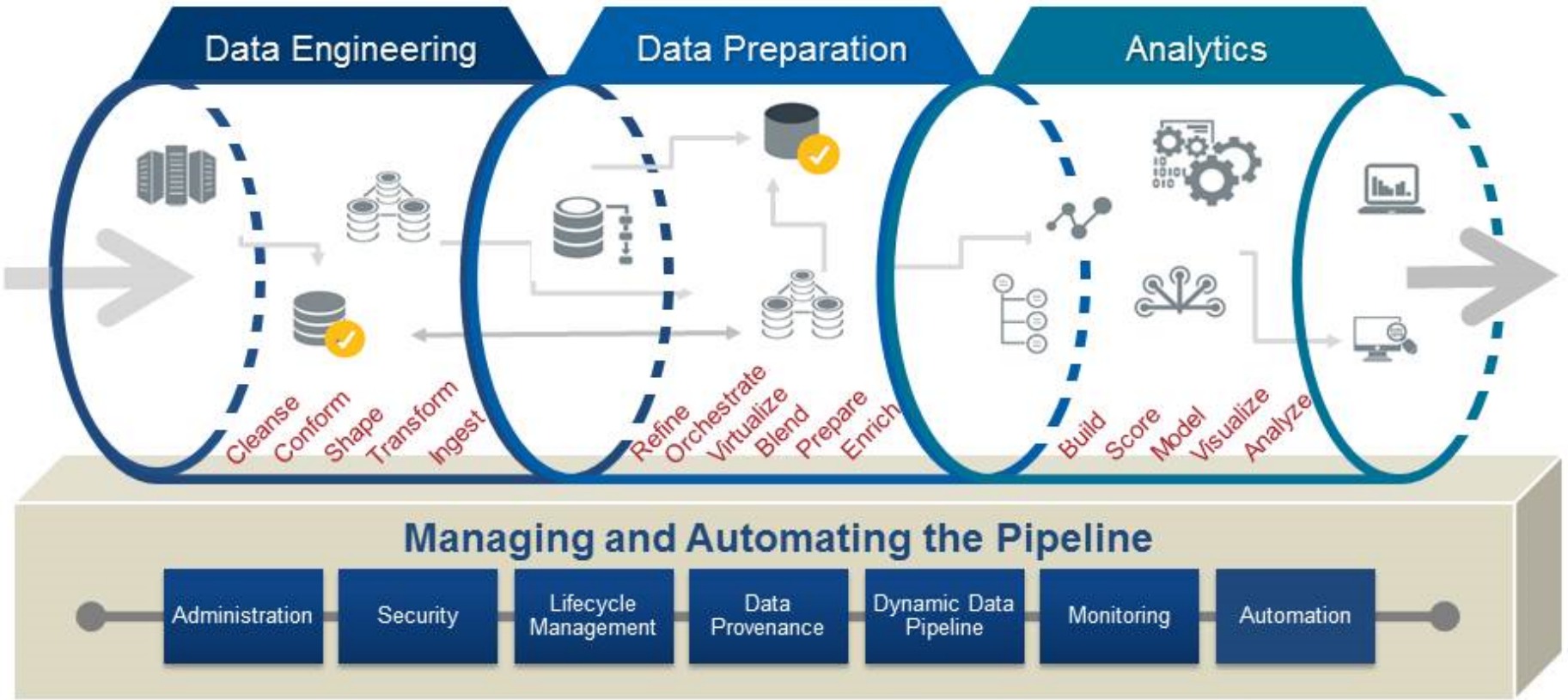
Analytics Pipeline (cont.)



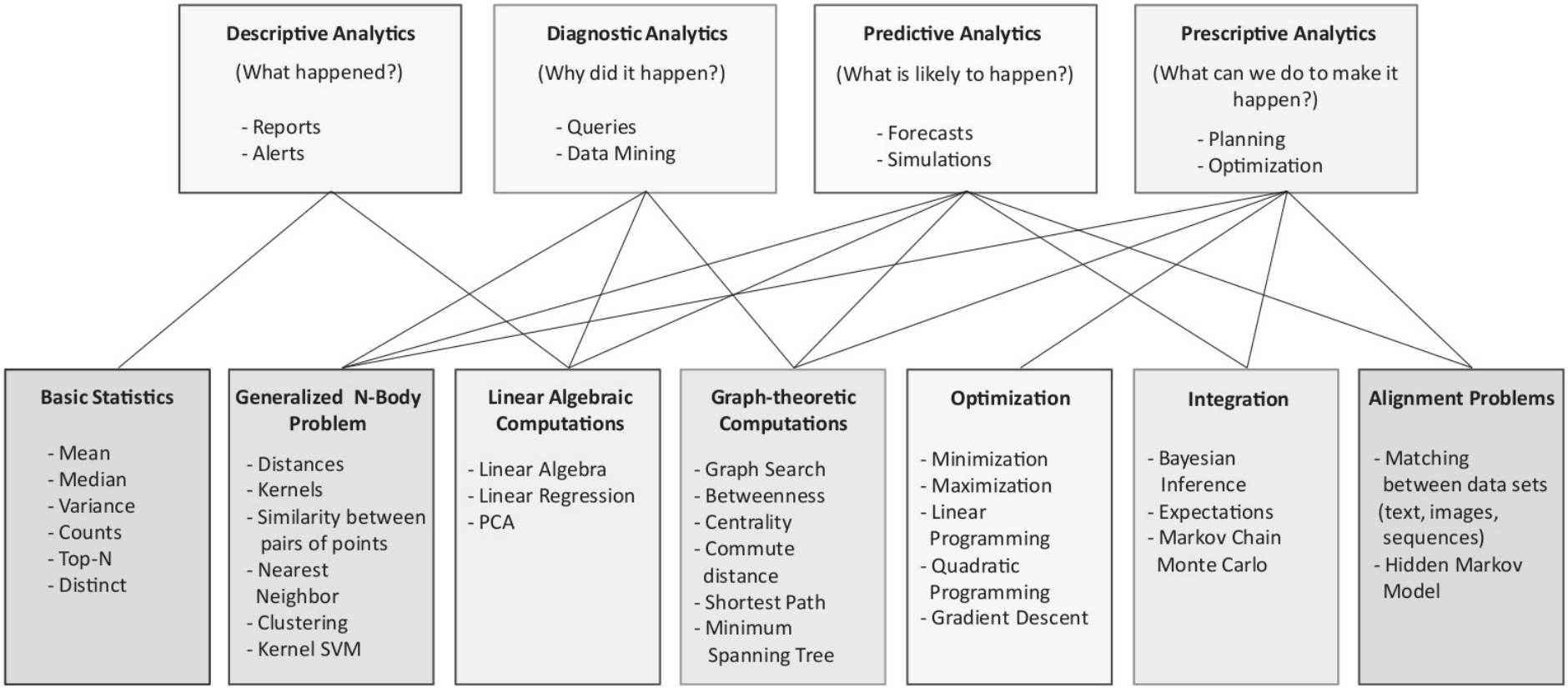
Analytics Pipeline (cont.)

Analysis Techniques

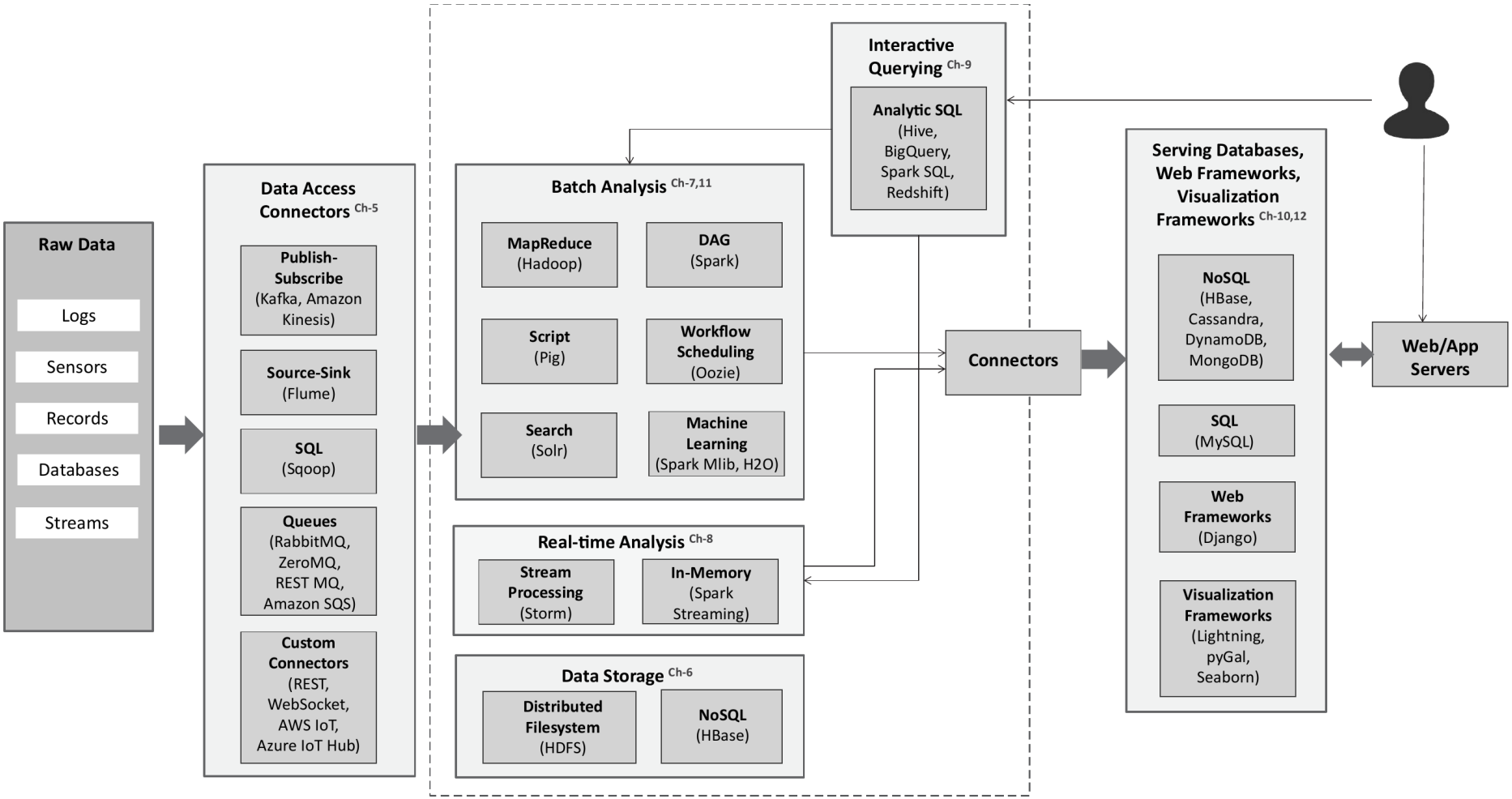
- Batch Analysis
- Real-Time Analysis



Analytics Pipeline (cont.)



Big Data Stack



Big Data Stack (cont.)

Hadoop

- Open Source Framework by Apache for Distributed Batch Processing
- Uses Map-Reduce Programming Model
- Data Stored on Hadoop Distributed File System (HDFS)
- Data processed in Hadoop Clusters
- Contains other tools for job scheduling, and machine learning
- Available directly from Apache, or Cloudera, or HortonWorks

Spark

- Open Source Framework by Apache for Distributed Parallel Processing
- Uses Map-Reduce Programming Model
- Data Stored in Memory (RAM)
- Visualizes operations by constructing Directed Acyclic Graphs
- 100s of times faster than Hadoop for sorting, machine learning operations

Overview

Hadoop common

For Storage

Hadoop File System

For Processing

Map Reduce

Implementations

- Cloudera
- Amazon
- Ali Baba
- or Direct Installation

Hadoop File System

Design Consideration

- Break files into blocks $B = \{b_1, b_2, \ldots, b_n\}$ of certain size s
- Distribute blocks across multiple (data) nodes $N = \{n_1, n_2, \ldots, n_m\}$ within a cluster
- Challenges:
  - For large m, node failures are quite probable. So introduce redundancy.
  - For large n, high throughput is required. So introduce concepts such as write once read many, and move computation closer to data.

Hadoop File System (cont.)

Performance Impact

- File Size, Block Length, Block Quantity, all affect performance
- Block quantity $\leftrightarrow$ Number of Threads
  - Queues
  - Thread Creation/Deletion Time
  - I/O
  - Messages exchanges between threads
- Strategies: Merge Files, Load Files in Sequence

Hadoop Common Architecture

Name Node (Master Server)

- Manages File System Namespace
- Regulates/Control access to files
  - Read/write requests from client
  - Create/Delete/Replicate blocks on data nodes

Data Nodes

- Manages Physical Storage of Blocks
- Serve Read/Write requests of Clients
- Serve Create/Delete/Replicate block requests of Name Node

Node Failures

Data Node Failure

- Server Crash / Disk Crash / Data Corruption / Network Failure
- Name node sends periodic heart-beats
- If true, mark dead, re-replicate block copy

Network Failure

- Denial of Service
- Physical Network failure

Namenode Failure

- Server Crash / Disk Crash / Data Corruption
- Send data/metadata to secondary nameserver (if configured)

HDFS Tuning Parameters

(Name,Value) properties in `/etc/hadoop/hdfs-site.xml`

- `dfs.replication` : 3 for Replication Factor
- `dfs.namenode.name.dir`: `/var/lib/hadoop/hdfs/name` for Name Node
- `dfs.datanode.data.dir`: `/var/lib/hadoop/hdfs/data` for Data Node
- `dfs.namenode.secondary.http-address` : `hdfs://localhost:50090` For Secondary Name node
- `dfs.permissions.superusergroup` : `hadoop` for User Permissions (must belong to this group)
- `dfs.block.size` : 134217728 for changing block size (across all clusters)
- `dfs.datanode.handler.count` : 10 for changing threads per data node.
- `dfs.namenode.fs-limits.max-blocks-per-file` : 100 for fixing maximum allowable blocks per file

- Dozens of other parameters specified in `hdfs-default.xml` (search online)

Specific Adjustments

- `hdfs dfs -D dfs.blocksize=134217728 -put test_128MB.csv /user`

HDFS Commands

- `hdfs dfs -ls /`
- `hdfs dfs -lsr /` ls with recursive display
- `hdfs dfs -du` Disk usage
- `hdfs dfs -dus` Disk usage summary
- `hdfs dfs -mv src dest`
- `hdfs dfs -rm xyz` Remove file or empty directory
- `hdfs dfs -rmr xyz` Recursive remove file or directory
- `hdfs dfs -put local remote`
- `hdfs dfs -get remote local`
- `hdfs dfs -cat file`
- `hdfs dfs -tail file`
- `hdfs dfs -head file`
- `hdfs dfs -chmod 777 file`
- `hdfs dfs -chown group file`
- `hdfs dfsadmin -report` Shows utilization of HDFS

Using C

libhdfs

- Part of the Hadoop distribution (Located pre-compiled in `$HADOOP_HDFS_HOME/lib/native/libhdfs.so`)
- Compatible with both Linux/Windows
- Java Native Interface (JNI) to Core Interface API in Java
- Thread Safe
- To compile (gcc)

```
gcc -I /opt/hadoop-3.2.1/include hdfsC.c
    -L /opt/hadoop-3.2.1/lib/native -lhdfs
    -L /opt/oracle-jdk-bin/jre/lib/amd64/server -ljvm|
```

- To run

```
CLASSPATH=$CLASSPATH:$(/opt/hadoop-3.2.1/bin/hadoop classpath --glob)
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/oracle-jdk-bin/jre/lib/amd64/server: \
                                 /opt/hadoop-3.2.1/lib/native
./a.out
```

Using C (cont.)

```c
#include "hdfs.h"
#include <string.h>
#include <stdio.h>

int main(int argc, char **argv) {
  hdfsFS      fs           = hdfsConnect("default", 0);

  const char  *writePath   = "/testfile.txt";
  hdfsFile    writeFile    = hdfsOpenFile(fs, writePath, O_WRONLY|O_CREAT, 0, 0, 0);

  char* buffer             = "Hello, World!";
  tSize num_written_bytes  = hdfsWrite(fs, writeFile, (void*)buffer, strlen(buffer)+1);

  hdfsFlush(fs, writeFile);
  hdfsCloseFile(fs, writeFile);
}
```

HTTP Rest API

- Support for HTTP Get, Put (-X PUT), Post (-X POST), Delete (-X DELETE)
- Configuration for `dfs.webhdfs.enabled` required in `hdfs-site.xml` (default is true)
- General Usage:

```
curl -i http://localhost:port/webhdfs/v1/[path|file]?
                                         [user.name=<user>&]
                                         op=[<options>]
```

- Default port: 50090 $\rightarrow$ 9870
- Responses in JSON

HTTP Rest API (cont.)

Sample Operations

- op=GETFILESTATUS Get Information about files
- op=MKDIRS for creating directory
- permission=755 for specifying Linux permissions
- op=CREATE for creating a (blank) file. For copying contents of an existing file, (use with -T <LOCAL_FILE>
- blocksize=<LONG> for Block Size
- replication=<SHORT> for replication factor
- op=APPEND for appending to a file
- op=OPEN for opening and reading a file
- op=RENAME&destination=<PATH> for renaming a file
- op=DELETE for deleting a file/directory, (Use with -X DELETE)
- . . . and others (See hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/WebHDFS.html

Map Reduce Framework

- Mapper Function
- Reducer Function
- Shuffle and Sort

Map Reduce Framework (cont.)

Example: Word Count

- <word, 1>
- Tokenize text and produce key-value pairs in a stream.
- Get new word from stream. If new word is same as previous word, increment a counter, else reset the counter.

```
import sys

for line in sys.stdin:
  line  = line.strip()
  words = line.split()
  for word in word:
    print ("%s\t1" % word)
```

```
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    count = int(count)
        if current_word == word:
        current_count += count
    else:
        print ('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word
```

Map Reduce Framework (cont.)

To run Job

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar
            -file /home/omar/work/codes/hadoop/mapper.py
            -file /home/omar/work/codes/hadoop/reducer.py
            -mapper mapper.py
            -reducer reducer.py
            -input /4300-folder/*
            -output /4300-output
```

- Reducer Threads controlled using `-D mapred.reduce.tasks=16`

Spark Overview

- Cluster computing platform designed for speed (mostly in-memory operations rather than file I/O)
- Support for Stream Processors (GPU's)
- API's available in Python, Java, Scala, and SQL

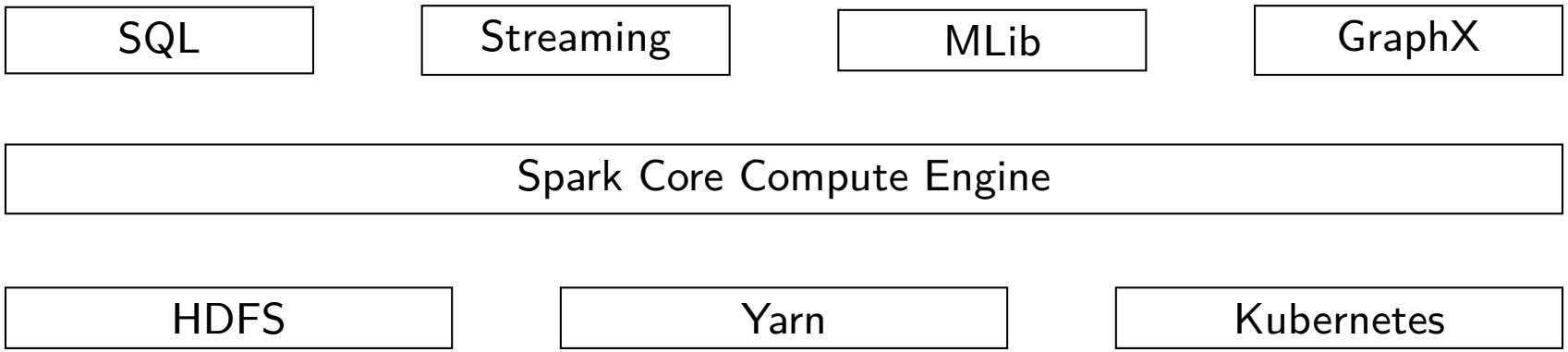
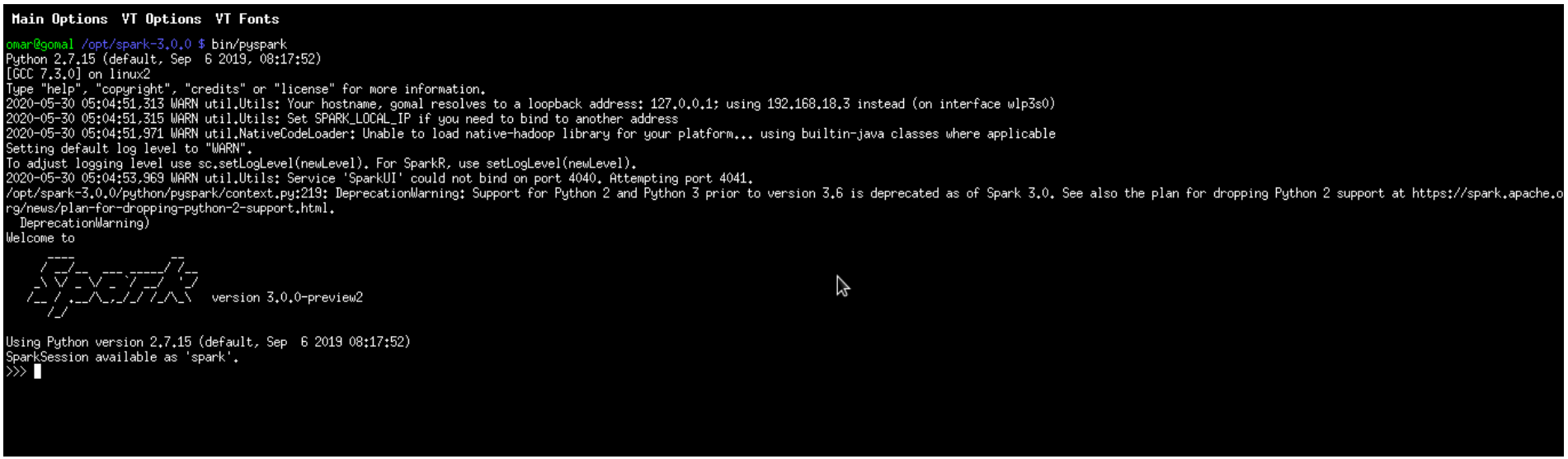


Figure 12: Spark System Architecture

- Web based Monitoring Interface on http://localhost:4040

Spark Shells

- `bin/pyspark` for Python
- `bin/spark-shell` for Scala
- `bin/sparkR` for R



- Can also be interfaced with Python Notebooks (e.g. Jupyter or iPython)

Hello World (Word Count)

```
lines = sc.textfile("README.md") # Must be on HDFS
                                 # lines is an RDD Object
lines.count()                    # Returns number of items
lines.first()                    # First line in RDD (or file)
```

Resilient Distributed Dataset RDD

- Computations performed on *distributed collections* that are automatically parallelized across a cluster.
- Immutable Objects (always new RDD returned)
- These Collections are the RDD's (more in coming slides)
- Created using `parallelize()` or `textfile()` methods of spark context.

Hello World (Word Count) (cont.)

Standalone Application

- Same API, but have to create Spark Contxt yourself
- Run using `bin/spark-submit`

```
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc   = SparkContext (conf = conf)

# User program from here onward

lines = sc.textFile("/4300-folder/4300-0.txt")
count = lines.count()

def hasBook(line):
    return "Book" in line

bookLines = lines.filter(hasBook)
bookcount = bookLines.count()
```

Core Spark Concepts

Driver Program (e.g., pyspark shell)

- Launches various parallel operations on a cluster
- Contains your applications main function
- Contains your applications distributed datasets
- Accesses Spark through a **Spark Context** object. This context is automatically created as object `sc`.

  ```
  <SparkContext master=local[*] appName=PySparkShell>
  ```
- RDD's created from Context

Executors

- Present on each Worker Node (computer)
- Managed by Driver

Core Spark Concepts (cont.)

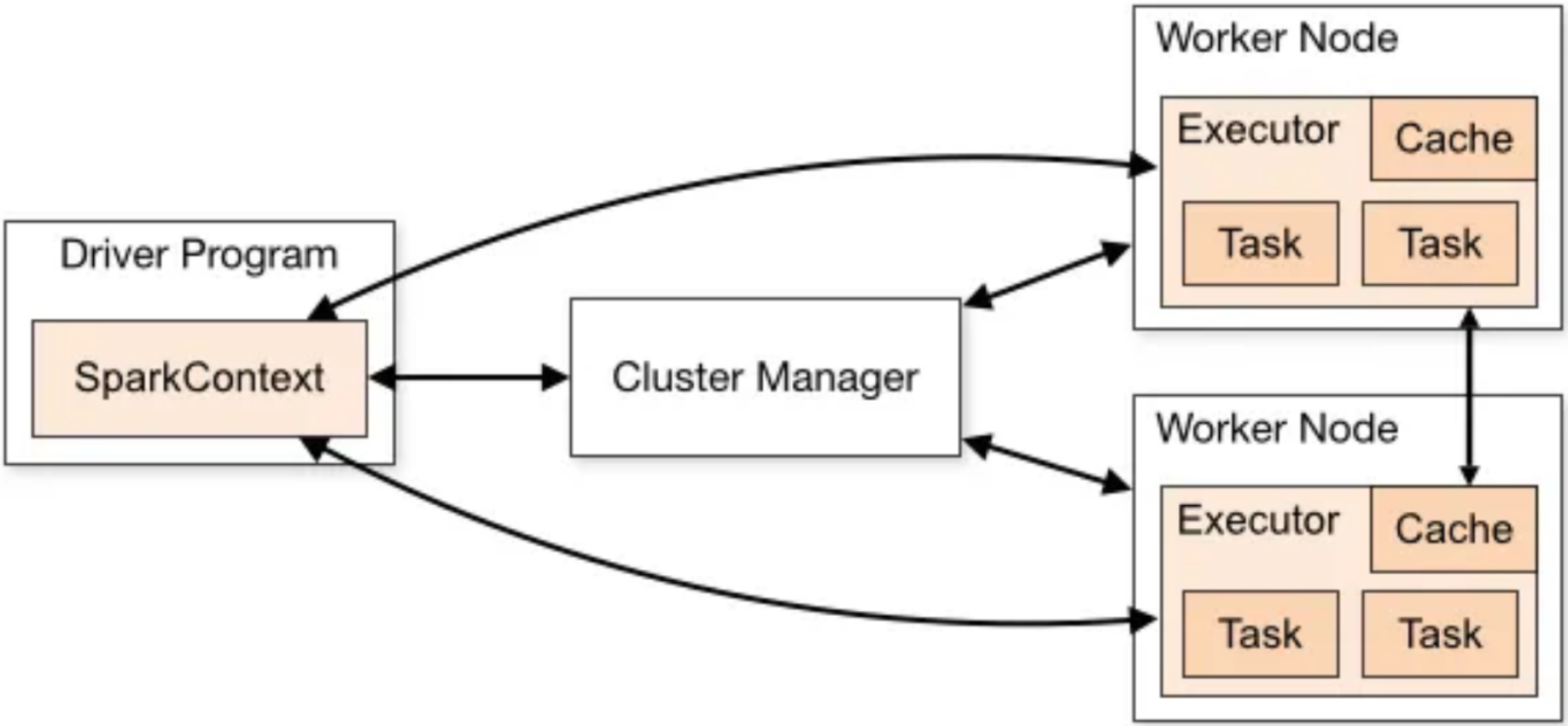


Figure 13: Components for Distributed Execution in Spark IMG: Apache Spark

Resilient Distributed Dataset (RDD)

Creation Approaches in Driver Program

- Loading an External Dataset from File `rddObj = sc.textFile("myFilename")`
- Collection of List objects `rddObj = sc.parallelize(["FAST", "I Like FAST"])`
  `rddObj = sc.parallelize([1, 2, 3, 4])`

Creation of DataFrame from RDD Object

```
dataCol = Seq("key", "value")
dataObj = Seq(("k1", v1), ("k2", v2))

rddObj  = sc.parallelize(dataObj)
dfRddObj = rddObj.toDF(dataCol)

dfRddObj.printSchema()
dfRddObj.show()
dfRddObj.select("key").show()
dfRddObj.filter(dfRddObj("value") > 100).show()
```

Resilient Distributed Dataset (RDD) (cont.)

Operations on RDD

- Actions (operations directly on RDD; Output displayed to Driver program, or to HDFS storage). For example: count(), first(), take(), collect()
- Transformations (new RDD from existing one). For example: Filtering(), Union(), Map(), FlatMap()

```
inputRDD  = sc.textFile("log.txt")

inputRDD.count()                                          # Action
inputRDD.first()                                          # Action

errorRDD   = inputRDD.filter(lambda x: "Error" in x)      # Transformation
warningRDD = inputRDD.filter(lambda x: "Warning" in x)    # Transformation

badLinesRDD = errorRDD.union(warningRDD)                  # Transformation

print("Bad Lines: " + badLinesRDD.count())                # Action
for line in badLinesRDD.take(10):                         # Action
      print(line)                                         # or file write
```

Resilient Distributed Dataset (RDD) (cont.)

```
for line in badlinesRDD.collect():                    # Danger Action
      print(line)
inputRDD  = sc.parallelize([1, 2, 3, 4])
squareRDD = inputRDD.map(lambda x: x * x).collect()    # Map Collect
for num in squareRDD:
      print("%d" % num)

inputRDD  = sc.parallelize(["Coffee Panda", "Happy Panda"])
outputRDD = inputRDD.map(lambda line: line.split(" "))
outputRDD.take(2)
# Displays: [['Coffee', 'Panda'], ['Happy', 'Panda']

outputRDD = inputRDD.flatMap(lambda line: line.split(" "))
outputRDD.take(4)
# Displays: ['Coffee', 'Panda', 'Happy', 'Panda']
```

Resilient Distributed Dataset (RDD) (cont.)

Lazy Loading Principle

- Compute/Retrive RDD only when required (determined through internal metadata)
- For transformation RDD's, maintain **Lineage Graph**
- Recompute again and again, any time you need it, with certain degree of caching (makes sense for large datasets). To override:

```
lines.persist()    # Hold data in memory
lines.count()
lines.first()
```

Other Transformation Operations

- `RDD1.distinct()` returns unique members
- `RDD1.union(RDD2)` returns Union
- `RDD1.intersection(RDD2)` returns Intersection
- `RDD1.subtract(RDD2)` returns RDD1 - RDD2
- `RDD1.cartesian(RDD2)` returns RDD1 $\times$ RDD2

Resilient Distributed Dataset (RDD) (cont.)

Saving RDD's (to HDFS)

- `lines.saveAsTextFile("Directory")`

Spark with Key Value Pairs

```
from pyspark import SparkContext, SparkConf

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf=conf)

words = sc.textFile("/documenttxt").flatMap(lambda line: line.split(" "))

wc    = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b : a +b)

wc.saveAsTextFile("/sparktest")
```

Vector Programming Overview

- Execution Models for GPU Architecture: MIMD, SIMD, SIMT

Figure 14: Fundamental Design Philosophy of CPU vs GPU

- **Control Logic**: Allow Parallel and/or Out-of-Order execution of threads. (Centralized on CPU, Decentralized on GPU)
- **ALU**: Perform arithmetic and bitwise operations (One ALU for each core on CPU, One ALU for each core on GPU, or One Arithmetic Unit (FPU) for each core)
- **Cache**: On-chip Memory to reduce instruction and data access latencies (Very small capacity on GPU)

Vector Programming Overview (cont.)

- **DRAM CPU**: Off-chip Memory to store different processes

Why are people switching to GPU's?

- Performance Reasons (Offload numerically intensive parts to GPU)
- Processor availability in Market (Program for the dominant processor. 10 years ago, parallel parallel computing limited to governments and large corporations/universities. This has all changed now with GPUs, thanks to video games.)
- Massive scalability in limited space (Embedded applications requiring parallelism could not include large cluster-based machines. With GPUs, they can)
- IEEE Floating Point Compliancy (Early GPU's were not entirely IEEE compliant. Hence programmers refrained to use them. This is now almost history, unless you buy an old GPU)
- Graphics Programming no longer required to operate on Graphics Cards. We have **GPGPU** compliant API's.

Example: NVIDIA Kepler K40

- GP-GPU, Scientific Computing
- Slave Processors
- GPU Giants (NVIDIA + AMD)
- CUDA: NVIDIA based GPU's
- OpenCL: Open coding standard for cross-device execution (Mobile Phones, GPU, CPU, Altera FPGA's), established by Khronos Group (2008)

-	Kepler K40	Intel i7-4900MQ
Cores	2,880	4 (8 Threads)
RAM	12 GB	-
Cache	48 Kb	8 Mb
Clock	876 MHz	2.8 GHz
Bwidth	288 GB/s	25.6 GB/s
FLOP (d)	1.40 TFLOP	13.97 GFLOP
FLOP (f)	4.29 TFLOP	25.76 GFLOP

NVIDIA k40, 2880 cores, 12 GB RAM

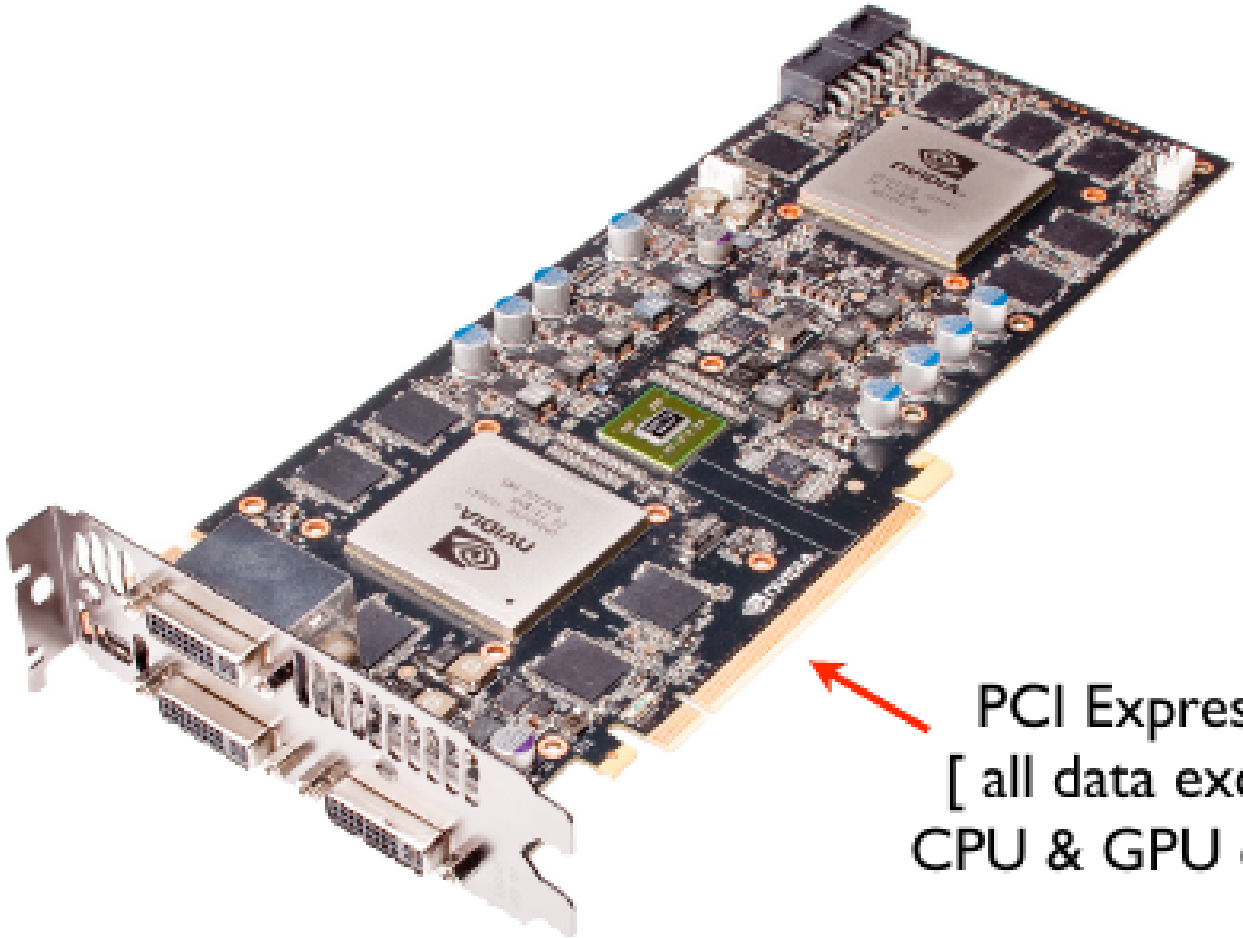
Example: NVIDIA GTX 590

GTX 590 GPU

GTX 590 GPU

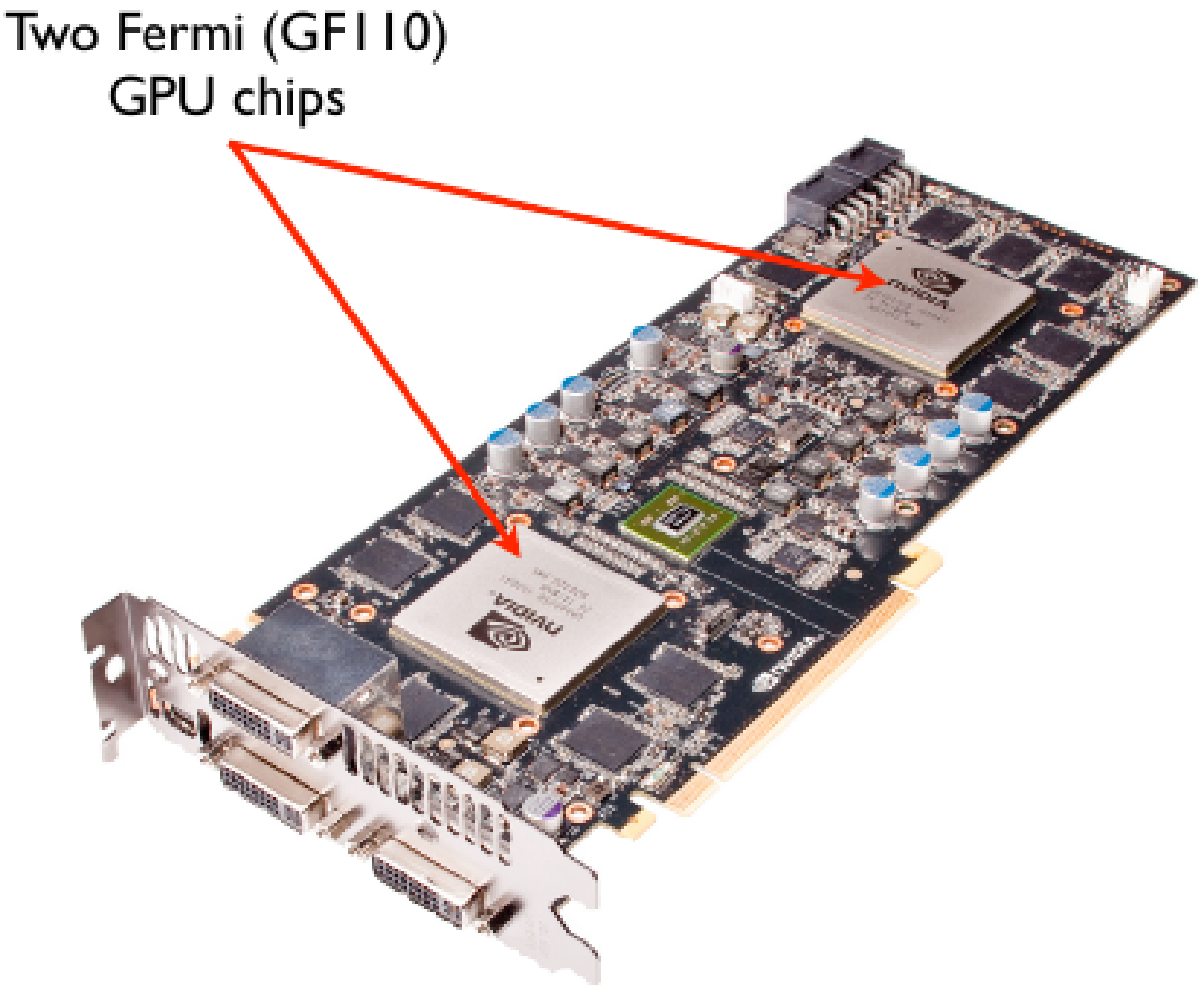
- Cores: 1024, Processor Clock: 1215 MHz, Memory: 3 GB

Example: NVIDIA GTX 590 (cont.)

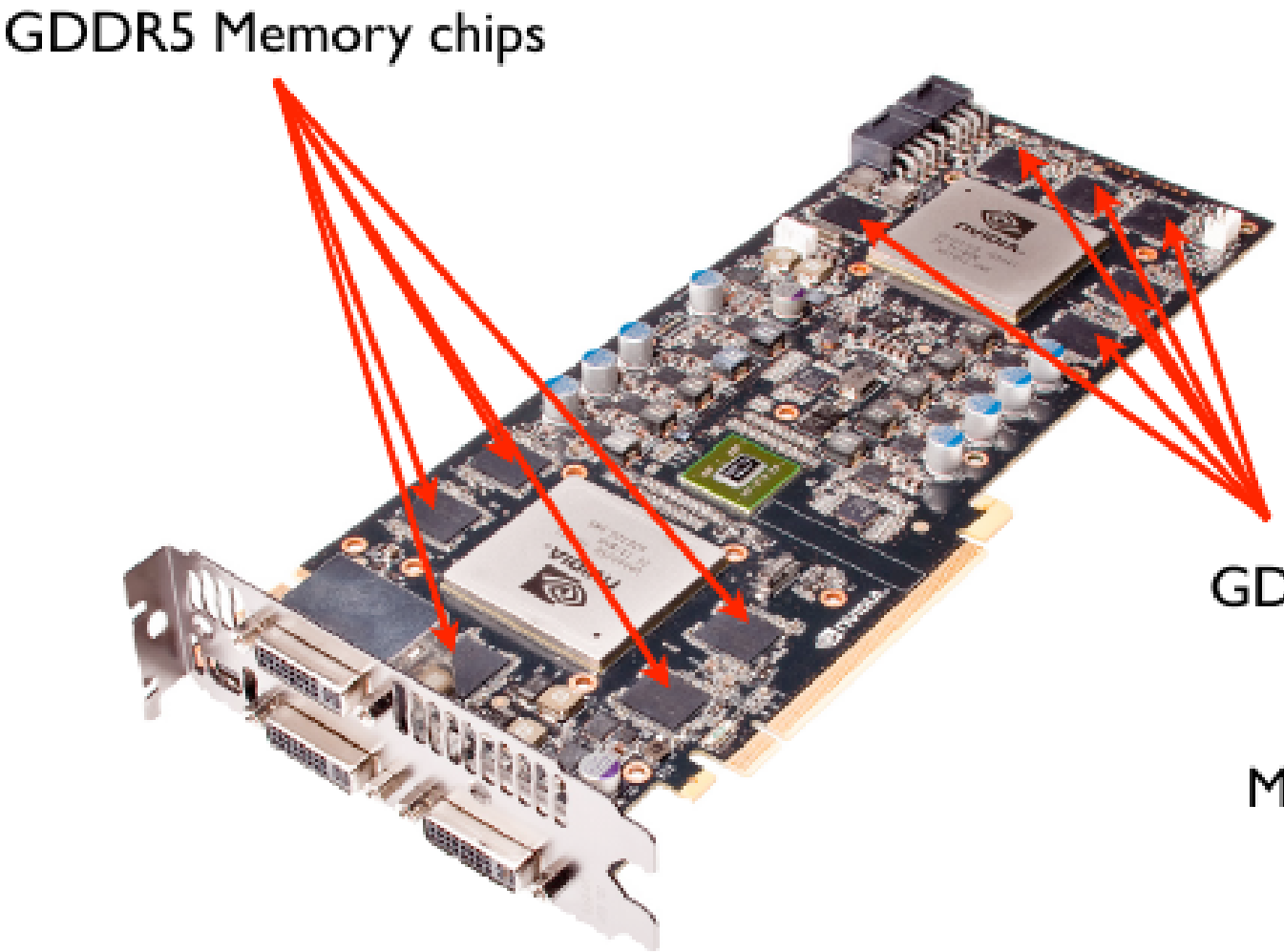


PCI Express 2.0 interface
[all data exchange between
CPU & GPU crosses this bus]

Example: NVIDIA GTX 590 (cont.)



Example: NVIDIA GTX 590 (cont.)



Example: NVIDIA GTX 590 (cont.)

Fermi: GPU <> GDDR5
memory bandwidth ~200 GB/s

PCI Express 2.0 bus
bandwidth ~6 GB/s

Xeon 5520 <> DDR3
memory transfer rate < 25 GB/s

Ethernet bandwidth:
~0.2 GB/s

USB3 bandwidth:
< 0.5 GB/s

SATA3 HD bandwidth:
< 0.8 GB/s

Figure 15: The Bottleneck

Trying it Out

Access Details

```
ssh 121.52.146.108 -l cdc-username -XY
```

where, **username** is your FAST-roll number
and **password** is same as username (one time only)
Example session with password = p116003 would be:

```
ssh 121.52.146.108 -l cdc-p116003 -XY
```

Introduction to GPGPU's

Open Computing Language (**OpenCL**)

- An **open** standard from **Khronos**; the makers of OpenGL (v1.0 Release December 2008)
- Cross Platform/Vendor/Architecture (CPU, GPU, DSP, FPGA, ...)
- GPU Giants (NVIDIA + AMD)
- Other Major Players (Apple, Intel, Qualcomm, Samsung, Xilinx, Altera)
- OpenCL is a {Standard, Language (based on C99), API/Library, Runtime SIMT based Compilation and Execution Environment}
- Two-way inter-operatable with OpenGL

Compute Unified Device Architecture (**CUDA**)

- **Proprietary** platform/API, released by **NVIDIA** (v1.0 released in January 2007)
- Handles only one platform/vendor, i.e., NVIDIA manufactured Graphic Cards
- CUDA is a {Language, API/Library, Non-runtime compilation environment, and Runtime execution environment}
- Inter-operatability with OpenGL is one-way (OpenGL can view CUDA buffers, but CUDA cannot view OpenGL buffers)

Introduction to GPGPU's (cont.)

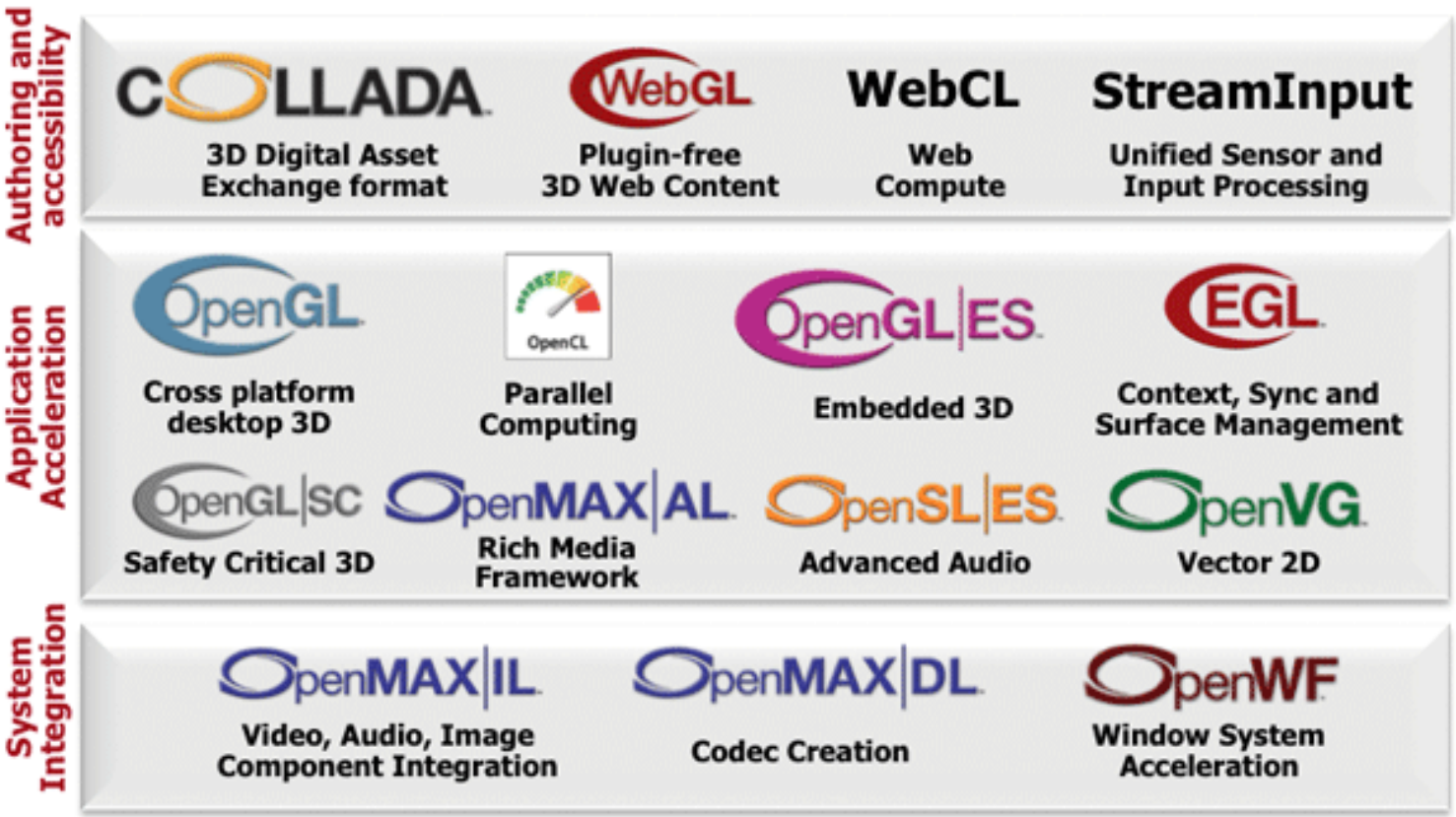
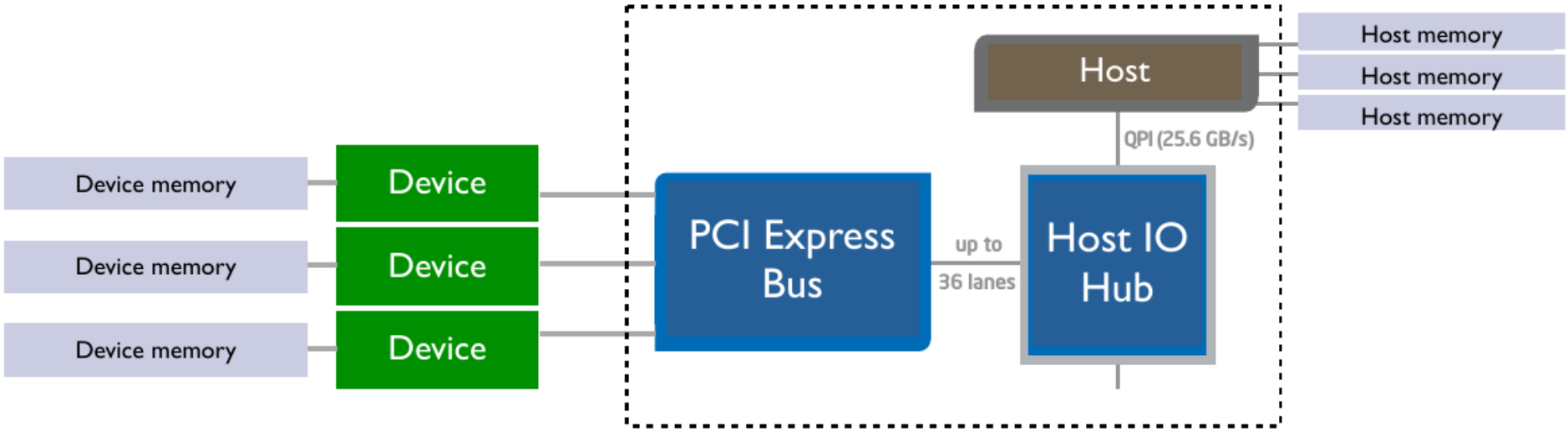


Figure 16: Khronos Group Open Specifications

[Img] http://www.khronos.org/about

Data Migration



Data Migration (cont.)

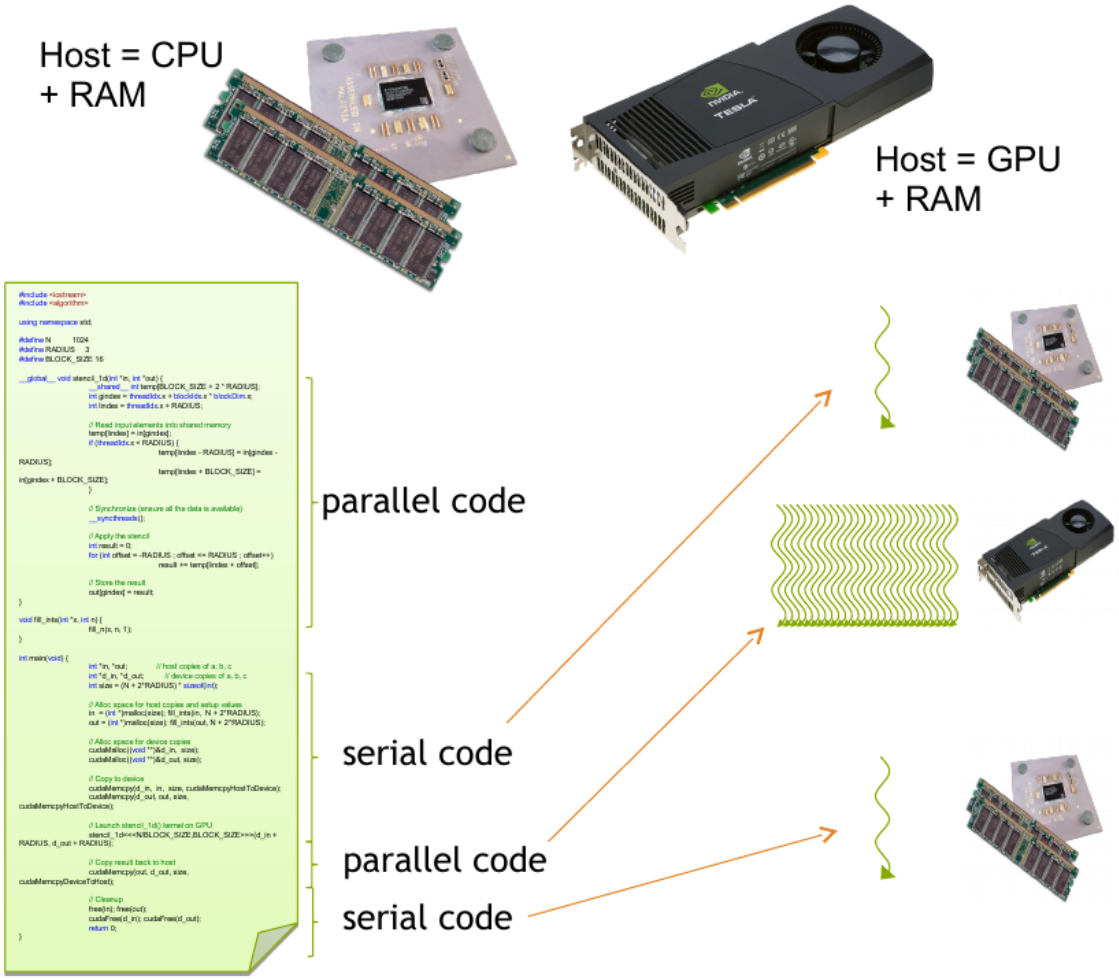
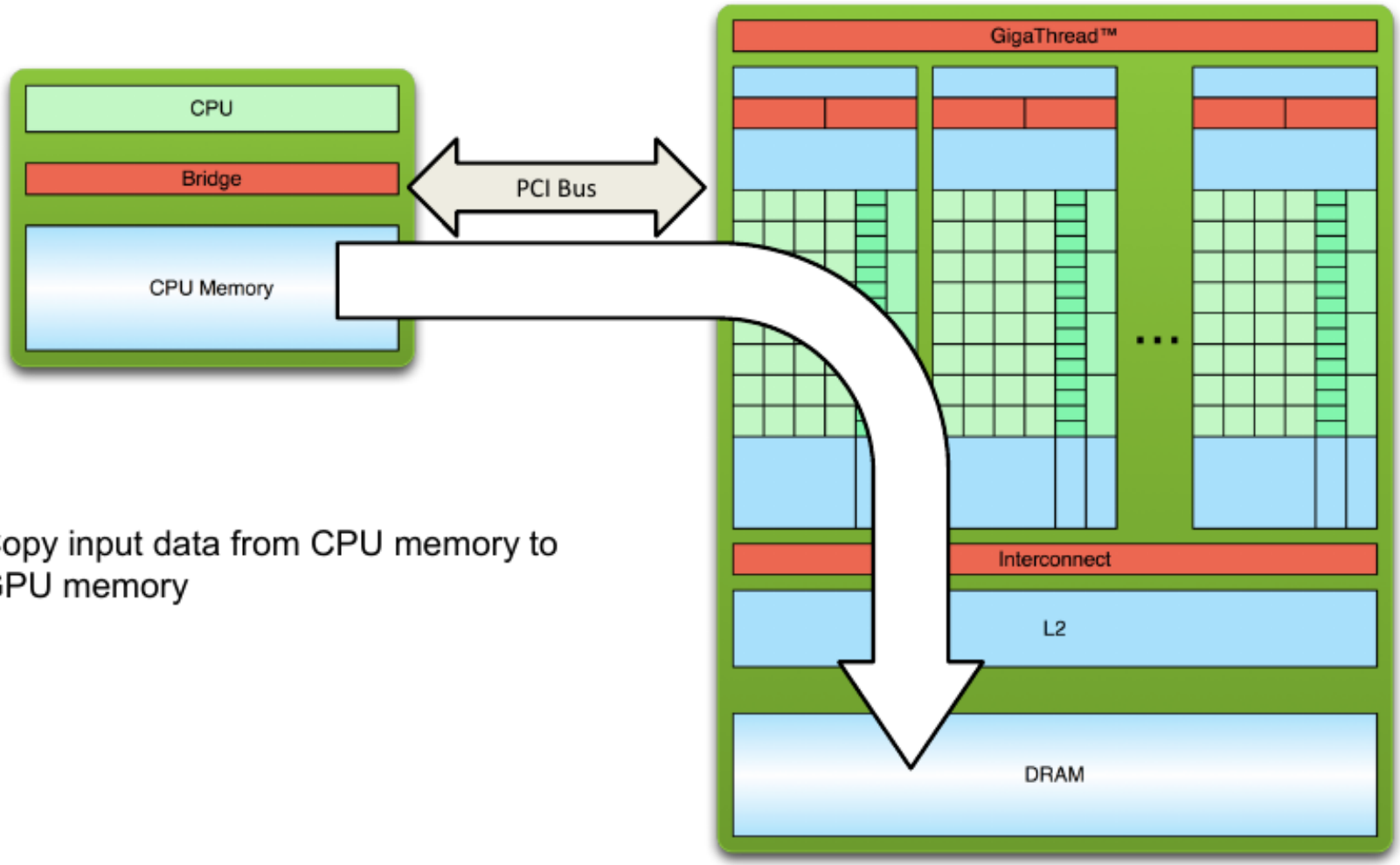


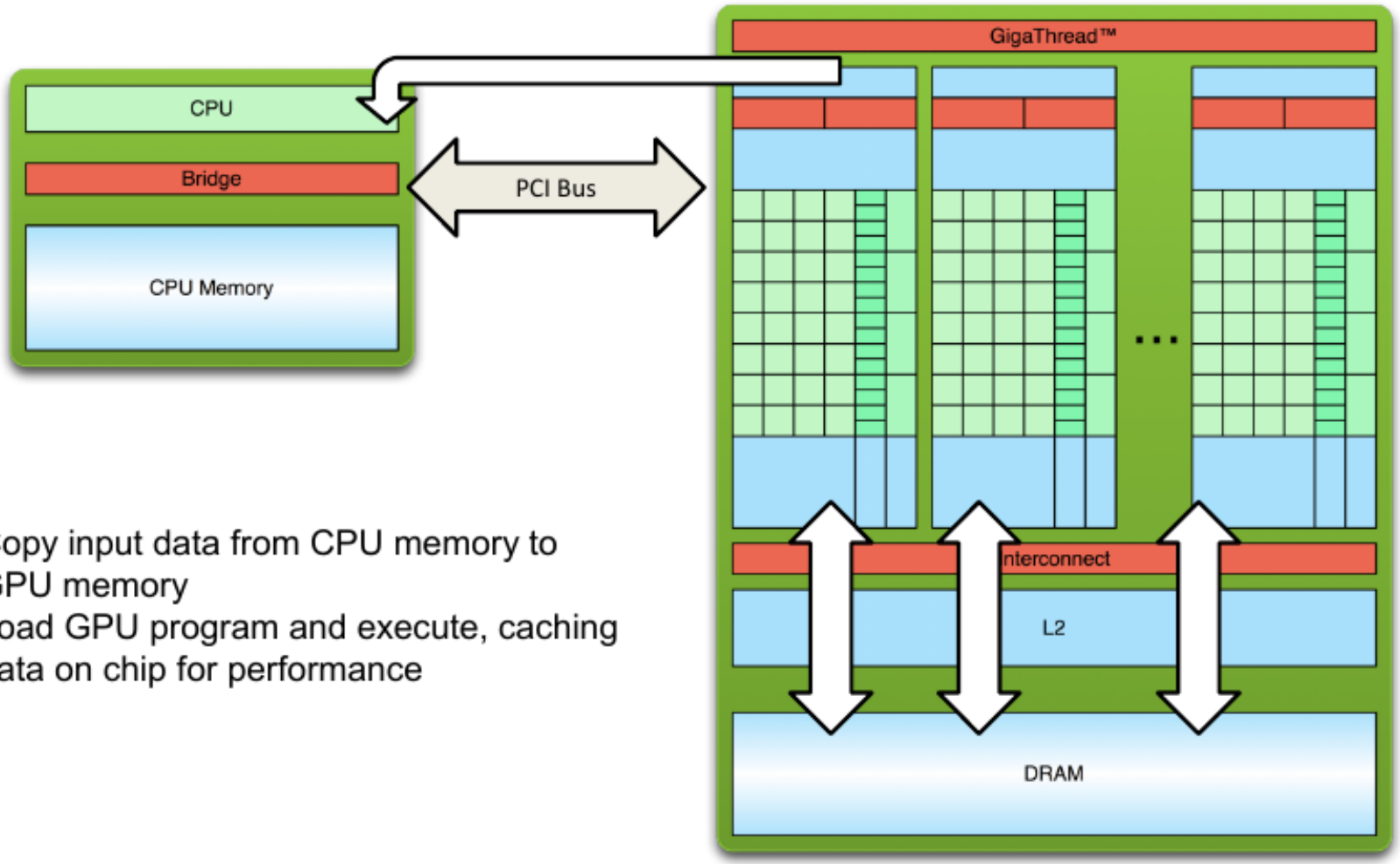
Figure 17: Porting portions of your code to GPU

Data Migration (cont.)



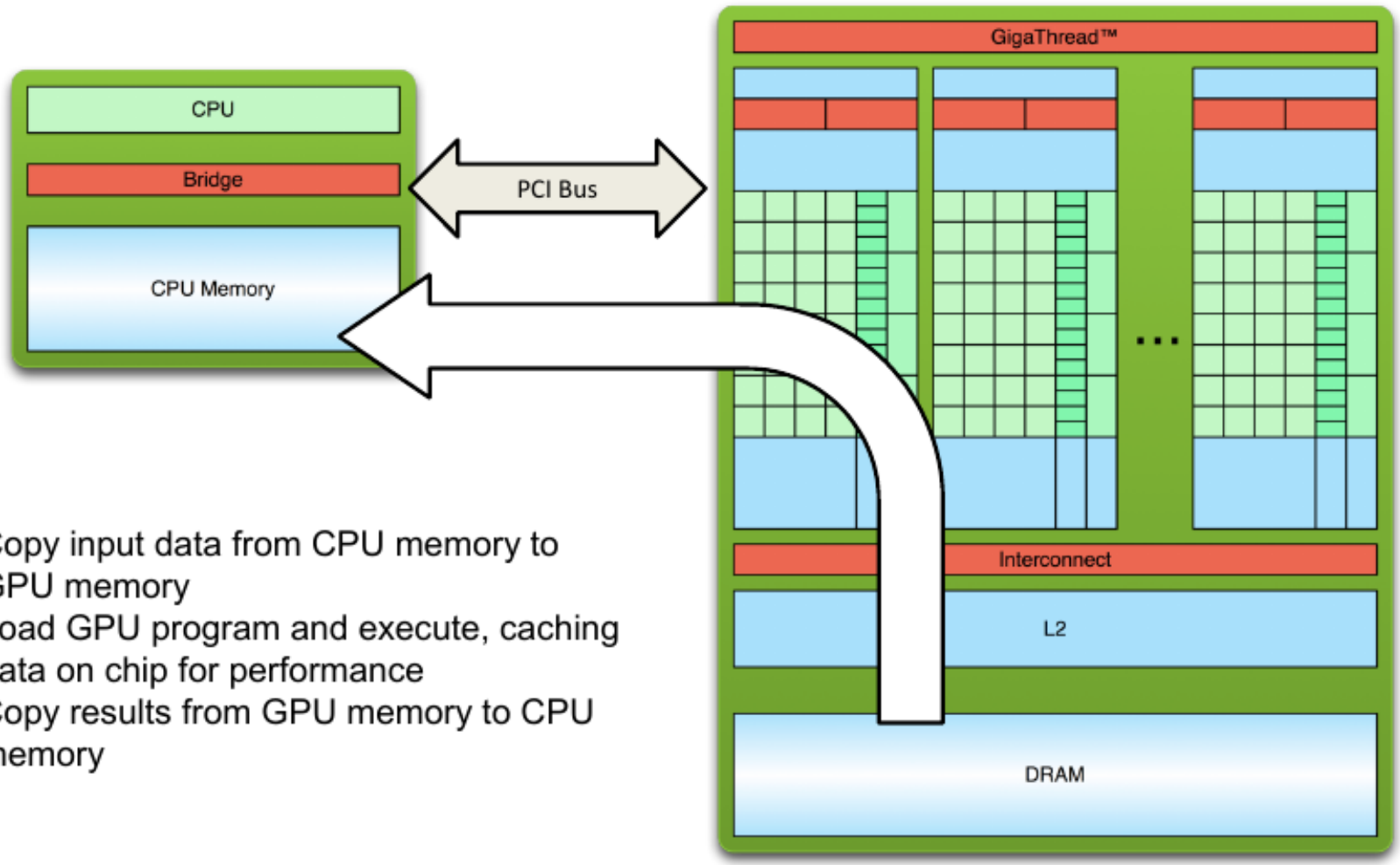
1. Copy input data from CPU memory to GPU memory

Data Migration (cont.)



1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance

Data Migration (cont.)



1. Copy input data from CPU memory to GPU memory
2. Load GPU program and execute, caching data on chip for performance
3. Copy results from GPU memory to CPU memory

Building up towards Hello World

```
int main()
{
   printf("hello world!\n");
   return 0;
}
```

Compilation

- nvcc hello_world.cu
- ./a.out

Building up towards Hello World: Inserting GPU Code

```
__global__ void myKernel(void)      // indicates function runs on
{                                   // device and is called from
}                                   // host code

int main()
{
   myKernel<<<1,1>>>();             // like function call, but
                                    // with more information
                                    // 1st digit number of blocks
                                    // 2nd digit threads per block
   printf("hello world!\n");
   return 0;
}
```

- nvcc will separate source code into host and device components
  - Device functions processed by NVIDIA compiler
  - Host functions processed by standard host compiler

```
/* a, b, c are pointers to memory location on the device */
__global__ void add(int *a, int *b, int *c)
{  *c = *a + *b;
}
int main()
{  int a=2, b=7, c, *da, *db, *dc;

   cudaMalloc((void **)&da, sizeof(int)); // allocate
   cudaMalloc((void **)&db, sizeof(int)); // on device
   cudaMalloc((void **)&dc, sizeof(int));

   // Copying in blocking-mode
   cudaMemcpy(da, &a, sizeof(int), cudaMemcpyHostToDevice);
   cudaMemcpy(da, &a, sizeof(int), cudaMemcpyHostToDevice);
   add<<<1,1>>>(da, db, dc);
   cudaMemcpy(&c, dc, sizeof(int), cudaMemcpyDeviceToHost);

   printf("a + b = %d\n", c);
   cudaFree(da); cudaFree(db); cudaFree(dc);
   return 0;
}
```

Running in Parallel

- so far; pointing parameters to GPU, and running single thread on GPU.
- Time to look at how to run things in parallel.

Code Change

```
// add<<<1, 1>>>(da, db, dc);
   add<<<N, 1>>>(da, db, dc);
```

- Instead of running add() once, execute it N times in parallel

Changes in Kernel Code

```
__global__ void add(int *a, int *b, int *c)
{
    c[blockIdx.x] = a[blockIdx.x] + b[blockIdx.x];
}
```

Changes in Host Code

```
int main()
{  int *a, *b, *c, *da, *db, *dc, N=5, i;

   a = (int*)malloc(sizeof(int)*N); // allocate host mem
   b = (int*)malloc(sizeof(int)*N); // and assign random
   c = (int*)malloc(sizeof(int)*N); // memory

   cudaMalloc((void **)&da, sizeof(int)*N);
   cudaMalloc((void **)&db, sizeof(int)*N);
   cudaMalloc((void **)&dc, sizeof(int)*N);

   cudaMemcpy(da, &a, sizeof(int)*N, cudaMemcpyHostToDevice);
   cudaMemcpy(da, &a, sizeof(int)*N, cudaMemcpyHostToDevice);
   add<<<N,1>>>(da, db, dc);
   cudaMemcpy(&c, dc, sizeof(int)*N, cudaMemcpyDeviceToHost);

   for (i = 0; i < N; i++)
       printf("a[%d] + b[%d] = %d\n", i, i, c[i]);
}
```

That was Blocks in Parallel. What about Threads in Parallel

Function Call Change

```
// add<<<1, 1>>>(da, db, dc); // single thread GPU
// add<<<N, 1>>>(da, db, dc); // N blocks on GPU
   add<<<1, N>>>(da, db, dc); // N threads on GPU
```

Changes in Kernel Code

```
__global__ void add(int *a, int *b, int *c)
{
    c[threadIdx.x] = a[threadIdx.x] + b[threadIdx.x];
}
```

- Rest of Host code would be the same

Blocks or Threads. Whatever. Code is running in Parallel anyways

- Let's look at sample specs for NVIDIA Kepler K40
  - Cores = 2,880
  - Multiprocessors = 15
  - Cores / multi-processor = 192
- If we make parallel blocks, 1 block will be assigned to 1 multi-processor. This means all multi-processors will be busy, but within the multi-processor, 1 core has work to do, the remaining others are free.
- If we make parallel threads and 1 block, only 1 multi-processor will be busy, the remaining will be free.
- In either case, the GPU is **under-utilized**. For maximum utilization, use both (blocks + threads)
- **Note: Above is programmer's interpretation. The actual execution model loosely follows this interpretation but has other restrictions also.**

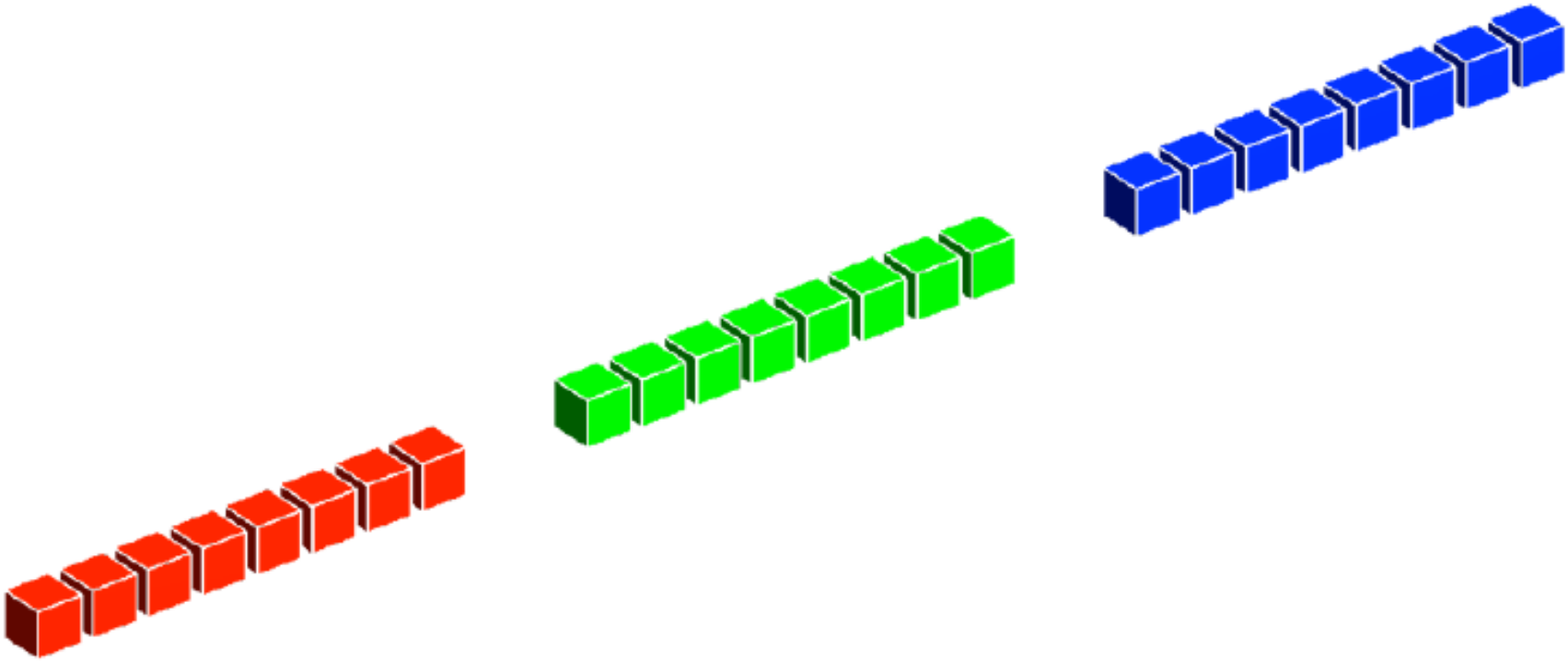
Case: Higher Dimensions

- So far, we have passed single values to kernel function call.
- For higher dimensions, use:
  - dimGrid(g_x, g_y, g_z) for dimensions of the grid
  - dimBlock(t_x, t_y, t_z) for dimensions of the thread block
- Total threads in thread-block cannot exceed 512 (or 1024 for some GPU's), i.e. condition $t_x \times t_y \times t_z \leq 512$ must be satisfied.
- Grid dimensions cannot exceed 32,768 in either dimension, i.e., condition $\max(g_x, g_y) \leq 32,768$
- dimGrid component must be evenly divisible by dimBlock component, i.e., condition $mod(g_n, t_n) = 0$ must be satisfied.

```
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS:3
CL_DEVICE_MAX_WORK_ITEM_SIZES:      1024 / 1024 / 64 (512/512/32)
CL_DEVICE_MAX_WORK_GROUP_SIZE:      1024 (512)
```

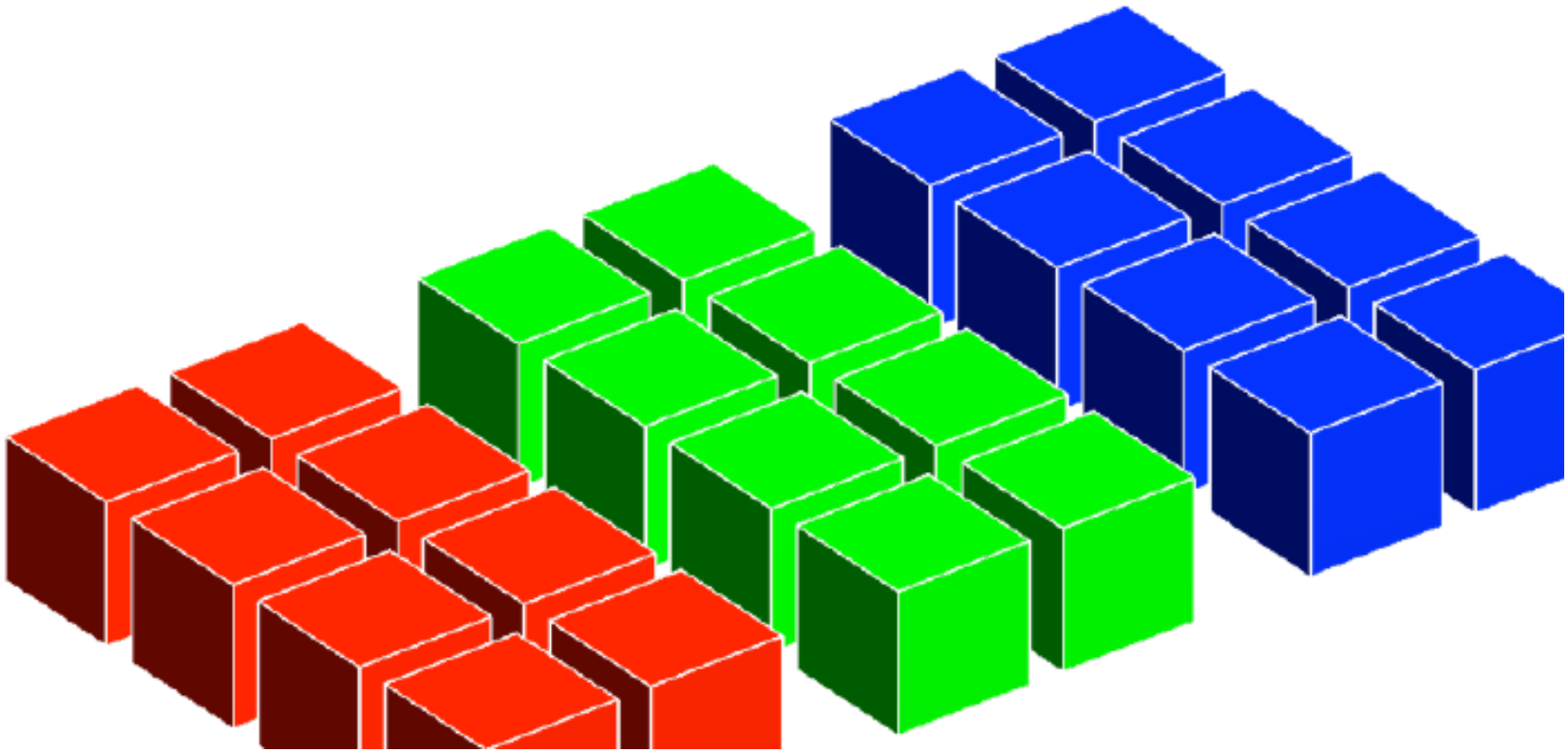
Case: Higher Dimensions (cont.)

```
dim3 dimGrid(3,1,1);
dim3 dimBlock(8,1,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



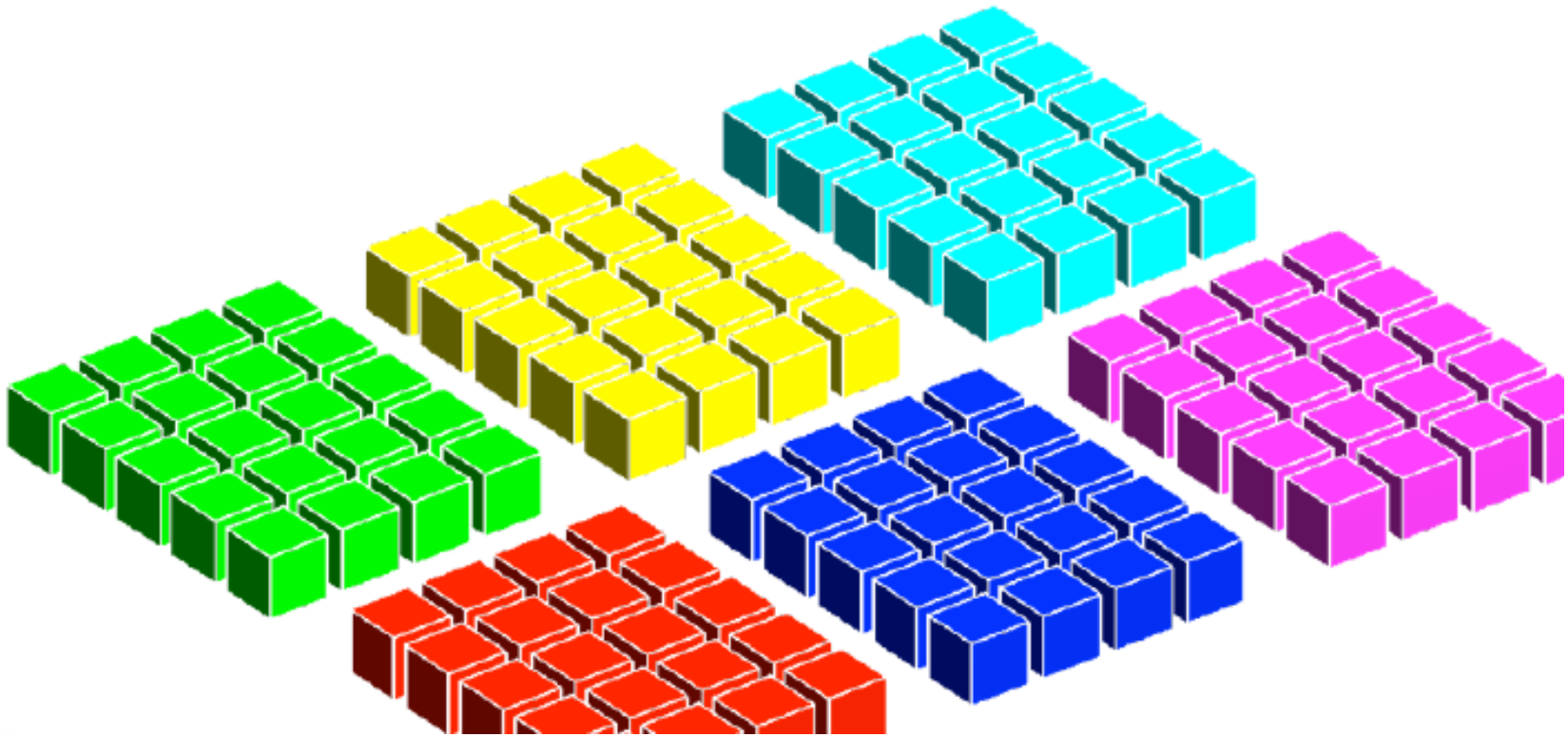
Case: Higher Dimensions (cont.)

```
dim3 dimGrid(3,1,1);
dim3 dimBlock(2,4,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



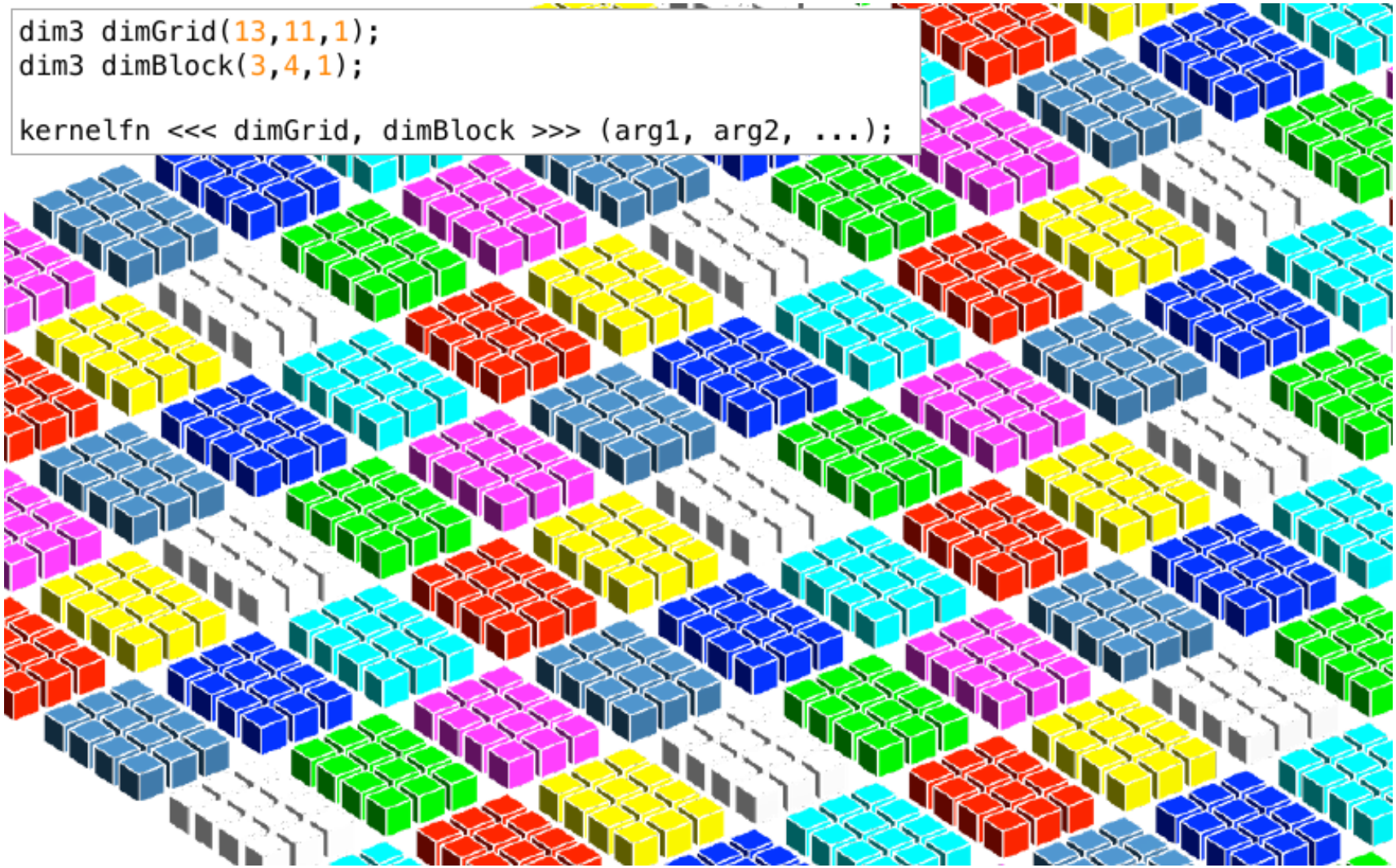
Case: Higher Dimensions (cont.)

```
dim3 dimGrid(3,2,1);
dim3 dimBlock(4,5,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



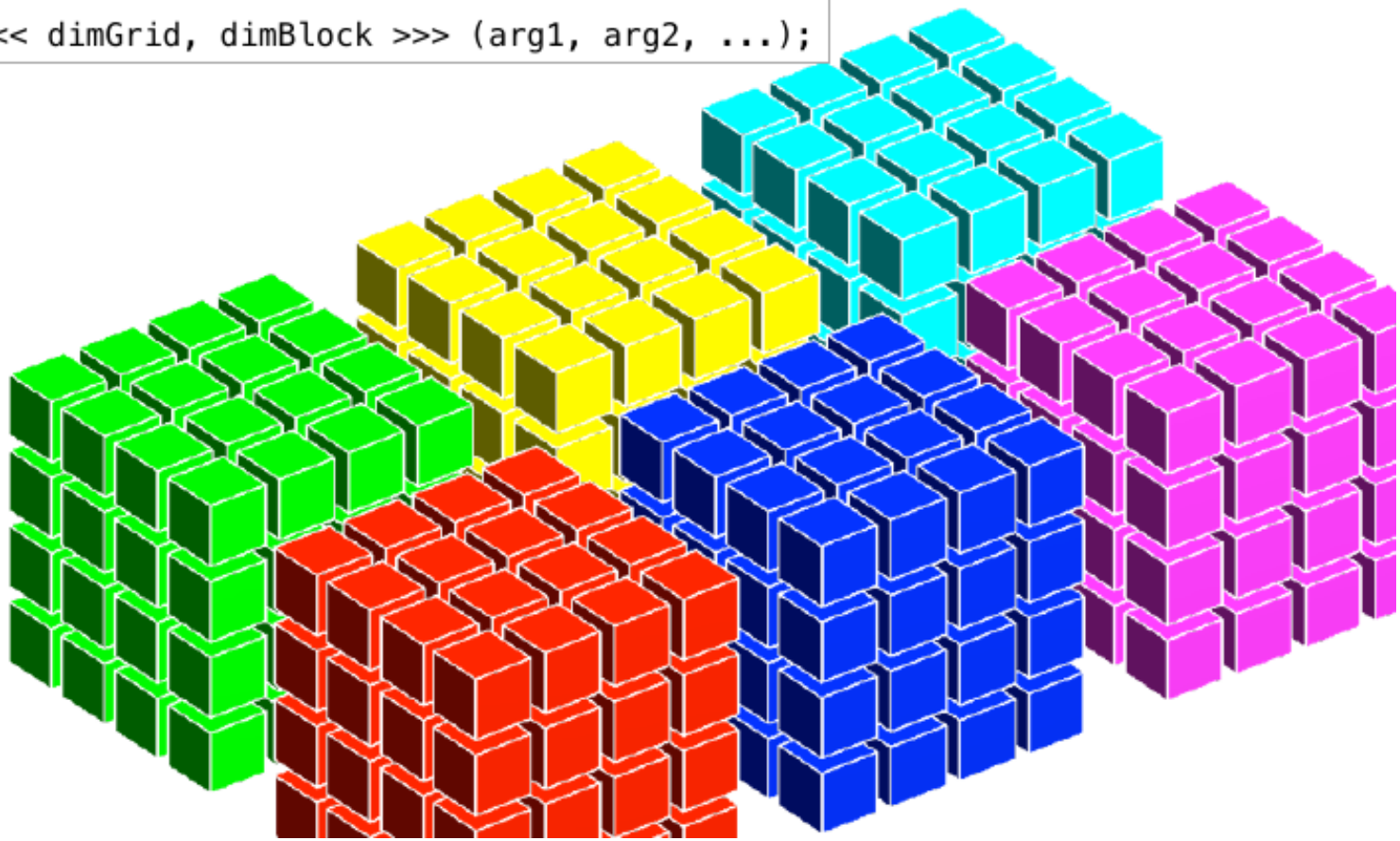
Case: Higher Dimensions (cont.)

```
dim3 dimGrid(13,11,1);
dim3 dimBlock(3,4,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



Case: Higher Dimensions (cont.)

```
dim3 dimGrid(3,2,1);
dim3 dimBlock(4,4,4);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



Combining both Concepts of Blocks and Threads

Formula to identify a point

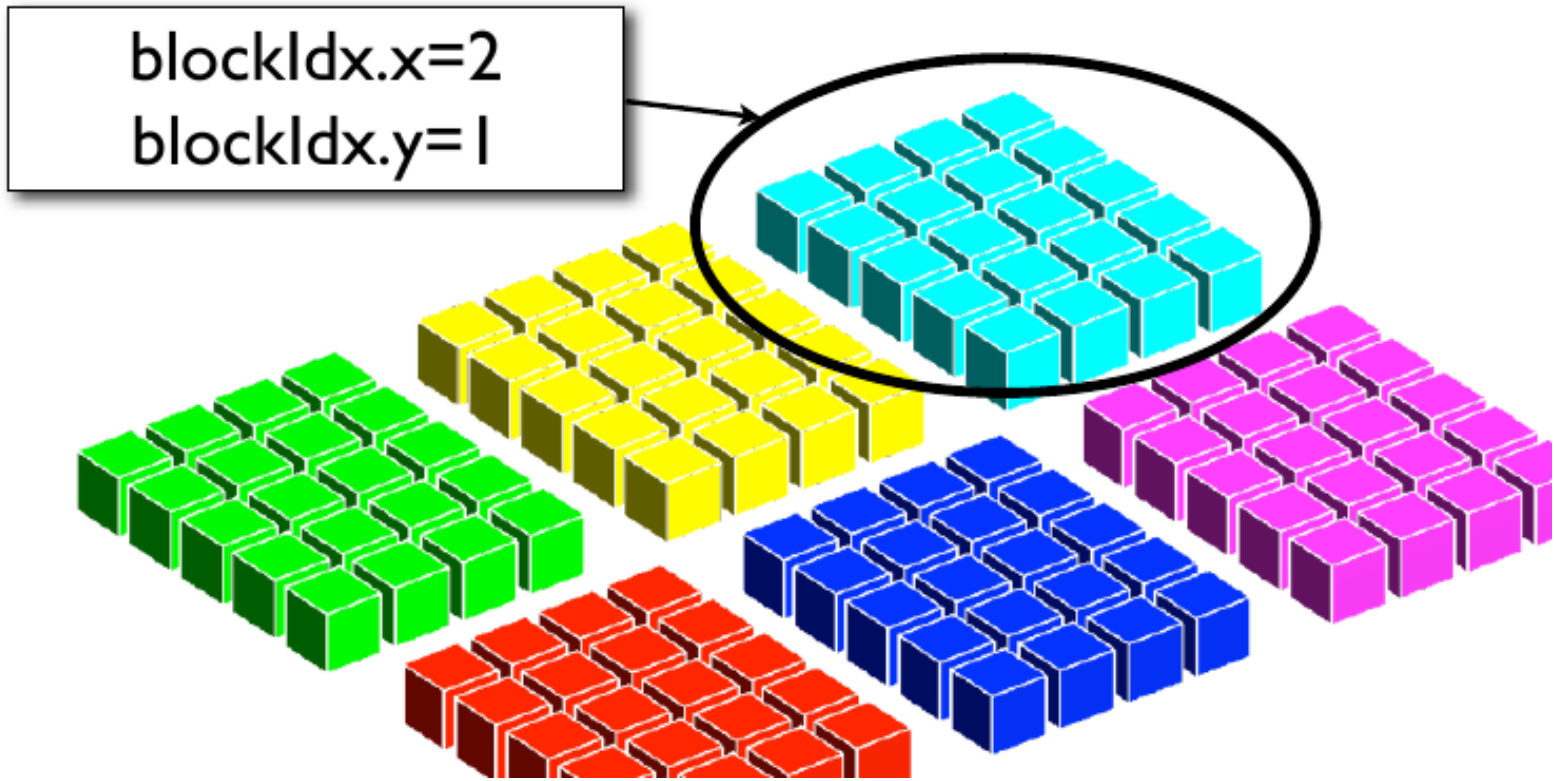
```
int index = blockIdx.x * blockDim.x + threadIdx.x;
```

- blockIdx.x = Index of thread block in grid
- blockDim.x = Number of threads in 1D thread-block
- threadIdx.x = Index of thread in 1D thread block

- Each thread execute kernel code is automatically provided the following variables
  - threadIdx.x, threadIdx.y, threadIdx.z
  - blockDim.x, blockDim.y, blockDim.z
  - blockIdx.x, blockIdx.y

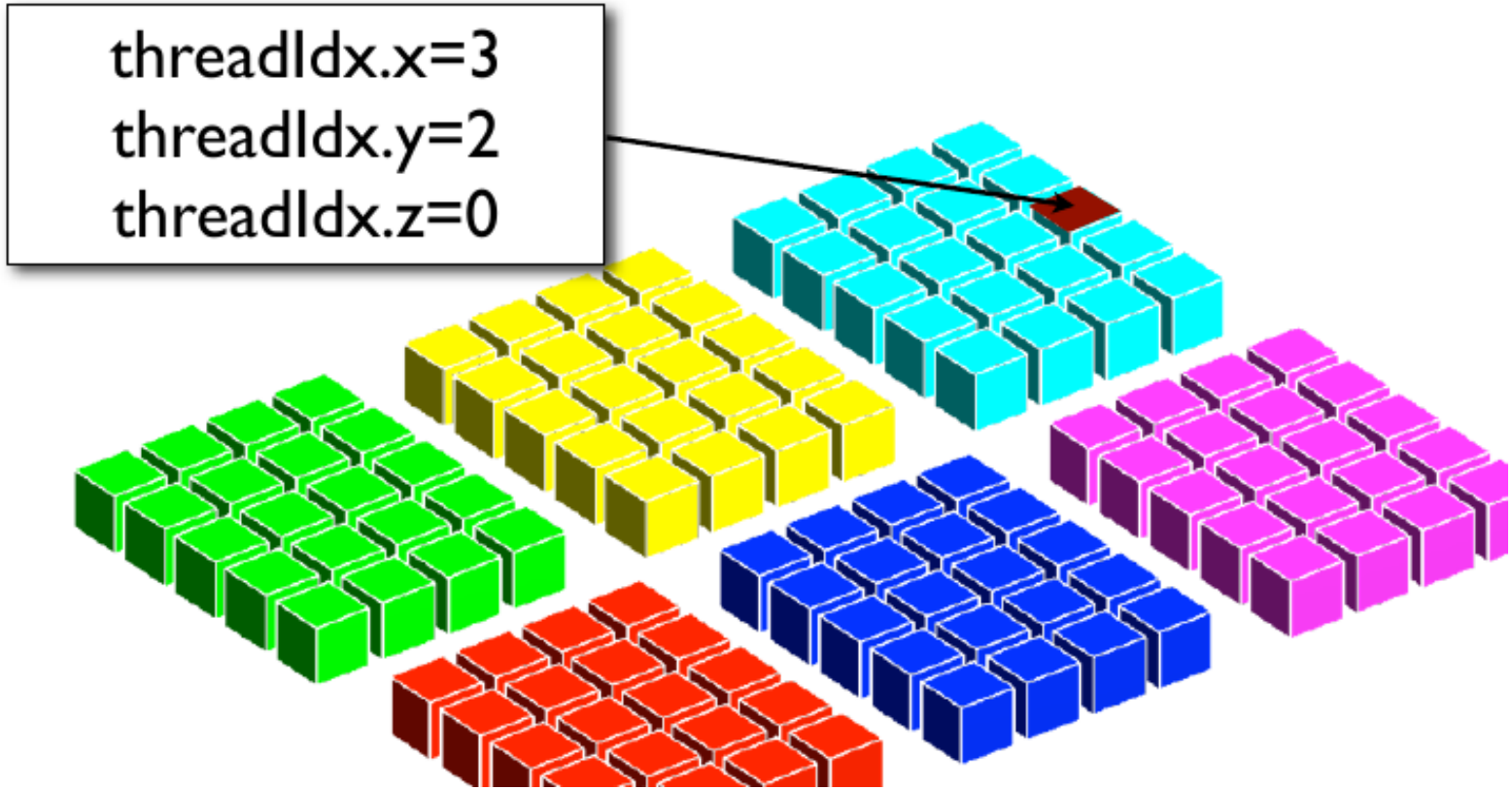
Combining both Concepts of Blocks and Threads (cont.)

```
dim3 dimGrid(3,2,1);
dim3 dimBlock(4,5,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```



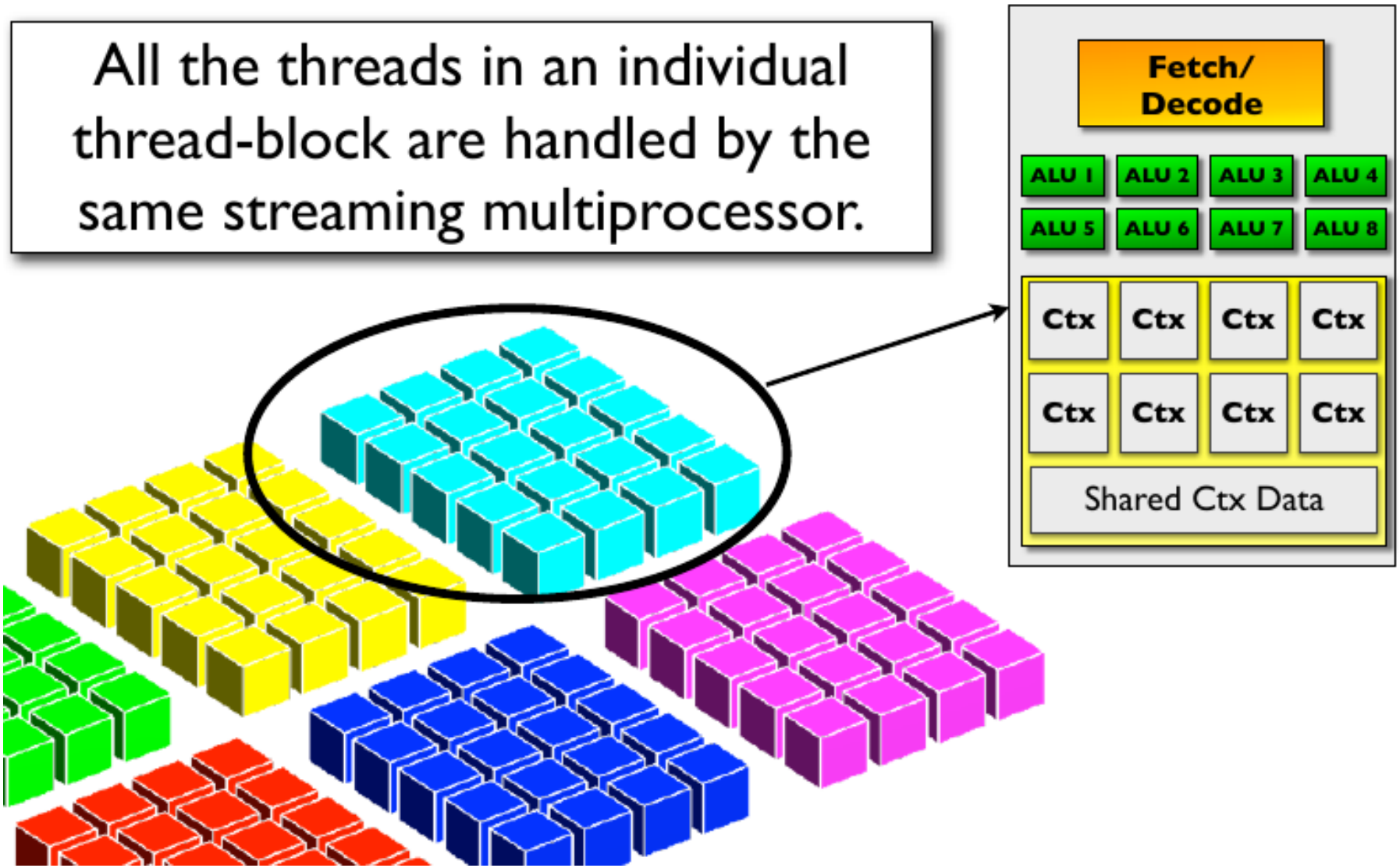
Combining both Concepts of Blocks and Threads (cont.)

```
dim3 dimGrid(3,2,1);
dim3 dimBlock(4,5,1);

kernelfn <<< dimGrid, dimBlock >>> (arg1, arg2, ...);
```

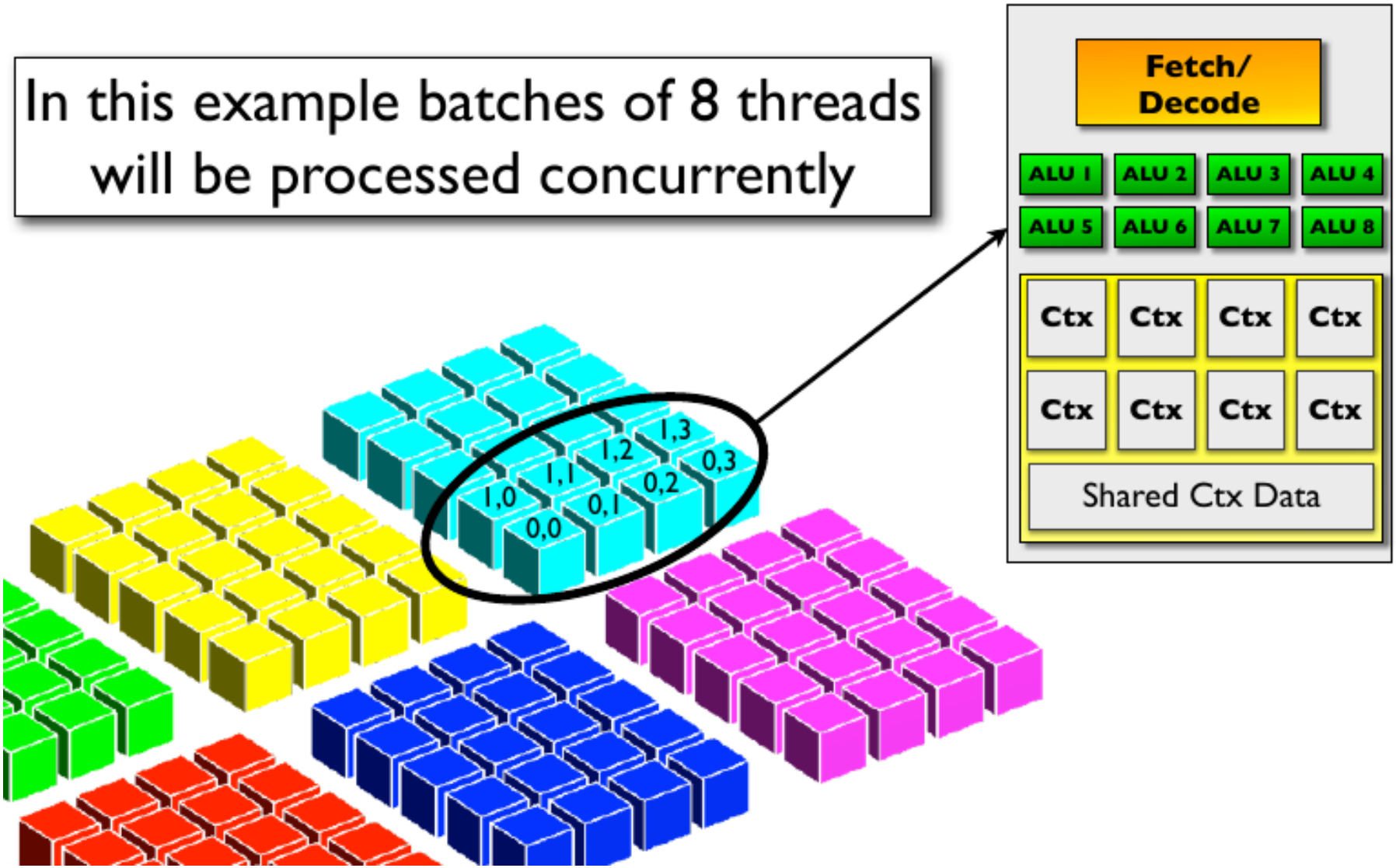


Higher Dimensions: Execution Model Revisited

All the threads in an individual thread-block are handled by the same streaming multiprocessor.



Higher Dimensions: Execution Model Revisited (cont.)



GPU Profiling using **nvprof**

- Provides textual reports on GPU and CPU activity

```
                nvprof ./a.out
ABCDEFGHIJKLMNOPQRSTUVWXYZ
==20580== NVPROF is profiling process 20580, command: ./a.out
25 25 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
==20580== Profiling application: ./a.out
==20580== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 42.82%  4.1280us         1  4.1280us  4.1280us  4.1280us  [CUDA memcpy HtoD]
 29.96%  2.8880us         1  2.8880us  2.8880us  2.8880us  square(char*)
 27.22%  2.6240us         1  2.6240us  2.6240us  2.6240us  [CUDA memcpy DtoH]

==20580== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 99.90%  604.38ms         1  604.38ms  604.38ms  604.38ms  cudaMalloc
  0.05%  318.97us        83  3.8430us     184ns  136.54us  cuDeviceGetAttribute
  0.02%  105.66us         1  105.66us  105.66us  105.66us  cudaFree
  0.01%  52.987us         1  52.987us  52.987us  52.987us  cuDeviceTotalMem
  0.01%  40.808us         2  20.404us  15.242us  25.566us  cudaMemcpy
  0.01%  36.906us         1  36.906us  36.906us  36.906us  cuDeviceGetName
  0.00%  16.190us         1  16.190us  16.190us  16.190us  cudaLaunch
  0.00%  2.7400us         1  2.7400us  2.7400us  2.7400us  cudaSetupArgument
  0.00%  1.6770us         2     838ns     357ns  1.3200us  cuDeviceGetCount
  0.00%  1.3360us         1  1.3360us  1.3360us  1.3360us  cudaConfigureCall
  0.00%     619ns         2     309ns     253ns     366ns  cuDeviceGet
```

Occupancy Calculator Usage

- Compile your program as:

  `nvcc myCode.cu --ptxas-options=-v`

  and you will see an output like:

```
ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z6squarePc' for 'sm_20'
ptxas info    : Function properties for _Z6squarePc
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 6 registers, 40 bytes cmem[0]
```

ptxas PTX Assembly Code (The GPU sees this after compilation. Think of it as machine readable code)

gmem Global Memory

spilling Occurs when there is no more space in registers

cmem Constant Memory, 0 for kernel arguments, 2 for user defined constant arguments, 16 for compiler generated constants

compute capability Device dependent. See ./oclDeviceQuery provided to you.

Occupancy Calculator Usage (cont.)

CUDA GPU Occupancy Calculator

Just follow steps 1, 2, and 3 below! (or click here for help)

1.) Select Compute Capability (click):	3.5	(Help)
1.b) Select Shared Memory Size Config (bytes)	49152	

2.) Enter your resource usage:		
Threads Per Block	256	(Help)
Registers Per Thread	6	
Shared Memory Per Block (bytes)	40	

(Don't edit anything below this line)

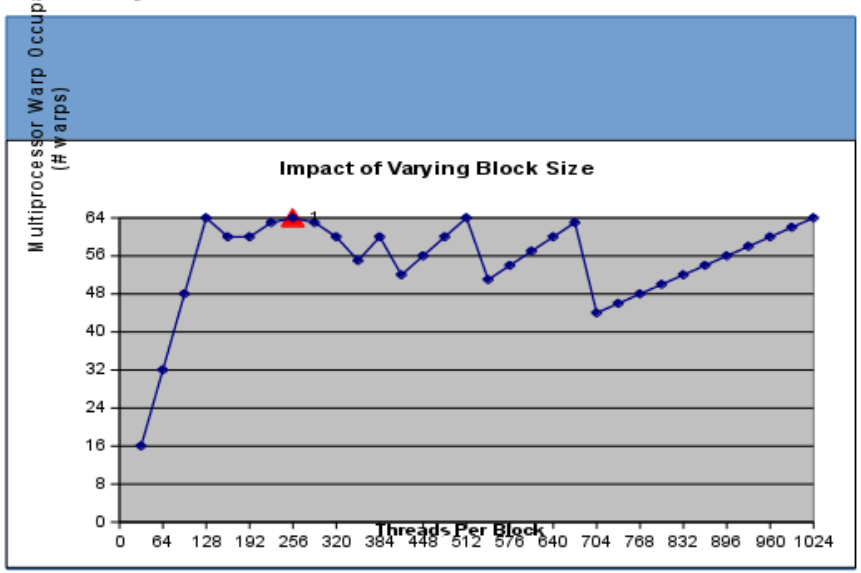
3.) GPU Occupancy Data is displayed here and in the graphs:		
Active Threads per Multiprocessor	2048	(Help)
Active Warps per Multiprocessor	64	
Active Thread Blocks per Multiprocessor	8	
Occupancy of each Multiprocessor	100%	

Physical Limits for GPU Compute Capability:	3.5
Threads per Warp	32
Warps per Multiprocessor	64
Threads per Multiprocessor	2048
Thread Blocks per Multiprocessor	16

lick Here for detailed instructions on how to use this occupancy calculator

For more information on NVIDIA CUDA, visit http://developer.nvidia.com/cuda

Your chosen resource usage is indicated by the red triangle on the graphs. The other data points represent the range of possible block sizes, register counts, and shared memory allocation.



Memory Coalescing

Code Snippet 1 (assume total threads = N)

```
int x = blockIdx.x * blockDim.x + threadIdx.x;
a[x] = a[x] * a[x];
```

Code Snippet 2 (assume total threads = N)

```
int x = blockIdx.x * blockDim.x + threadIdx.x;
if (x == N - 1)
    a[0] = a[x] * a[x];
else
    a[x] = a[x+1] * a[x+1];
```

- Looking at the hardware counters (NVIDIA Visual Profiler)
- Note: Will be more relaxed with higher Compute Capabilities on new graphic cards (Latest: 3.5)

Memory Coalescing (cont.)

Code Snippet 1

```
GLB MEM throughput(GB/s): 20.38
Peak for GT525M (GB/s): 28
Global M excess load(\%): 0.00
Global M excess store(\%): 0.00
```

Code Snippet 2

```
GLB MEM throughput(GB/s): 18.56
Peak for GT525M (GB/s): 28
Global M excess load(\%): 7.79
Global M excess store(\%): 0.00
```

Memory Coalescing (cont.)

Coalescing Rules

- Compute Capability 1.0 and 1.1
  - If 16 **aligned** threads require:
    - 4 bytes each, then 16 threads fetch 1 64 byte segment
    - 8 bytes each, then 16 threads fetch 1 128 byte segment
  - else if **not aligned** and require:
    - 4 bytes each, then 16 threads fetch 16 32 byte segment
    - 8 bytes each, then 16 threads fetch 16 32 byte segment
- Compute Capability 1.2 and 1.3
  - Prepare to Fetch 1 128 byte segment (for 16 threads)
  - If only lower/upper half of 128 byte is used, fetch 1 64 byte segment
  - If only lower/upper half of 64 byte is used, fetch 1 32 byte segment
- Compute Capability 2.x onwards
  - Fetch memory segment equal to size of cache

Pinned Memory

- Upto two times faster memory transfers.
- Uses concept of page-locked memory (prevents memory identified by pointer to be paged-out)
- Downside: Big matrices cannot be allocated using this approach.

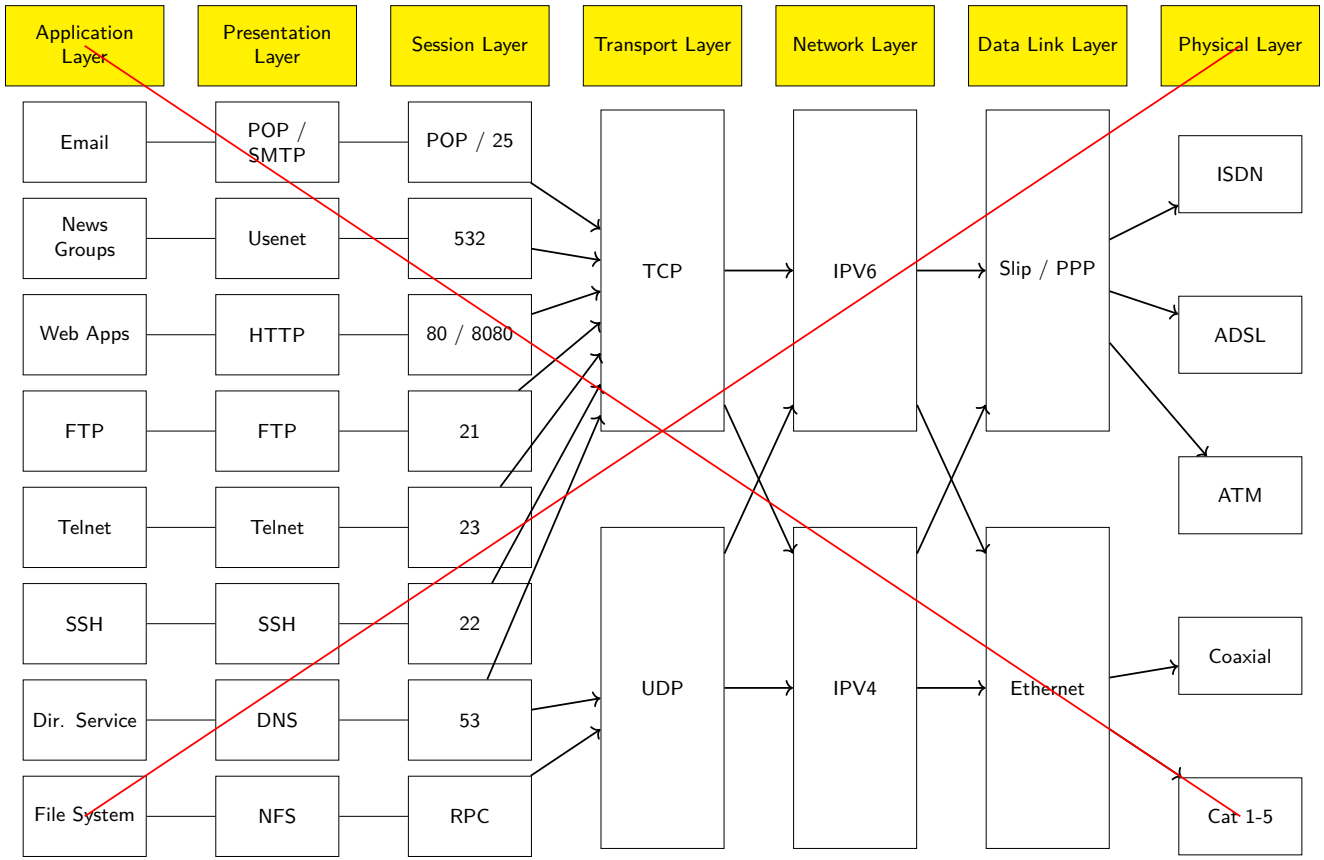
```
cudaHostAlloc((void **)&hMemory, sizeof(hMemory),
                              cudaHostAllocDefault);
```

Example Code (Measured using **valgrind –tool=pagein**)

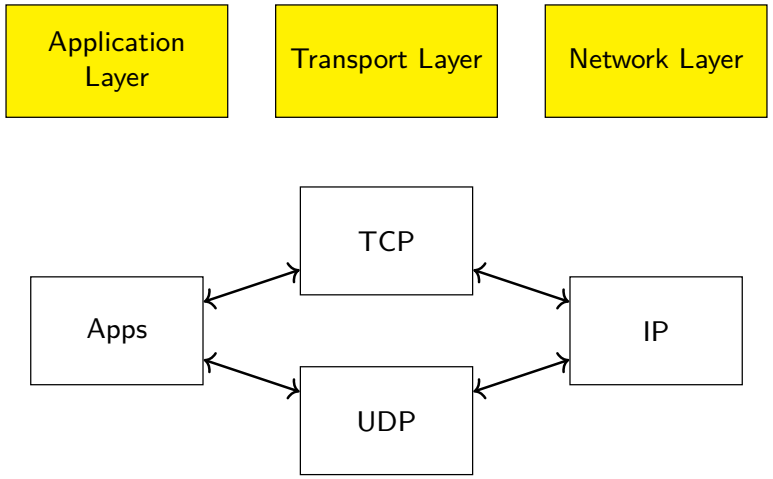
	Total	.data	.text	other
Pinned	137,447	4,594	129,502	4000
N. Pinned	248,143	114,597	129,502	4000

Client/Server Model

- Client: Request a Service
- Server: Provide a Service (Models: Iterative/Concurrent/Distributed)



Client/Server Model (cont.)



Overview

Info Required for Connection

- Protocol (TCP/UDP)
- Source + Destination IP Address
- Source + Destination Port Address

Transmission Control Protocol (TCP)

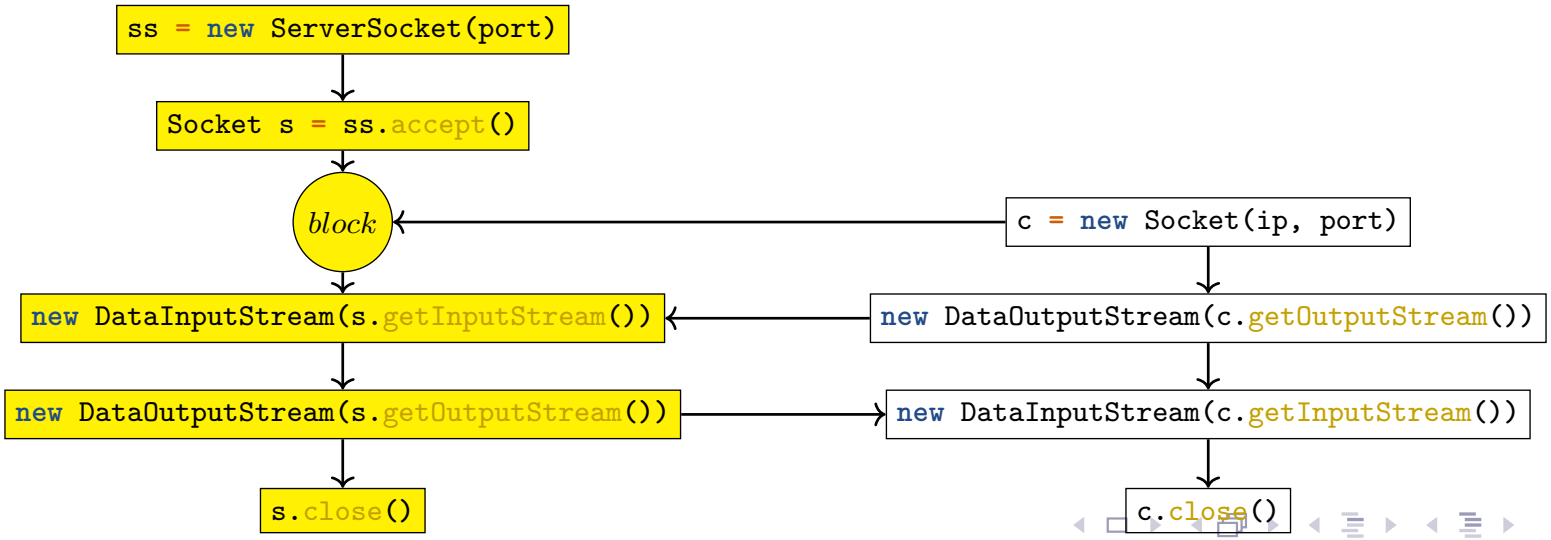
- Streams of Packets sent (each packet numbered for sequence)
- Acknowledgement sent for Each Packet (else retransmit)
- Packets also undergo error detection/correction techniques
- Slower, at the cost of integrity

User Datagram Protocol (UDP)

- Information content wise, datagram is same as packet (no numbering, packets can be received in any order)
- No acknowledgements sent
- No error detection/correction performed
- Faster, at the cost of integrity

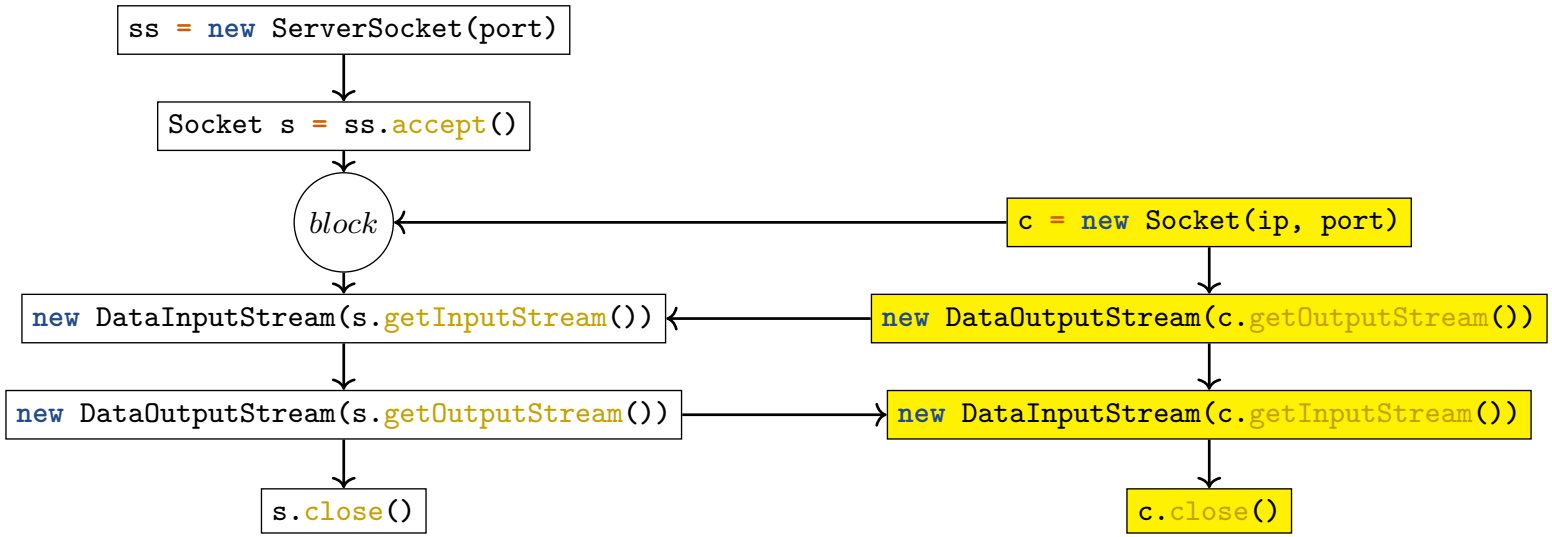
TCP Socket

- A method for achieving Inter-process communication on Linux systems (extended to networks)
- Person A phones Person B. Once connection is established. They can talk. Same Concept as Socket
- Just as File Read/Write through File Descriptor, similarly, Socket Send/Receive through File Descriptor as well
  - `java.net.Socket` and `java.net.ServerSocket` for Sockets
  - `java.io` for File Descriptors
- Most sockets use Half-Association
  - Protocol + Local IP Address + Local Port Number
  - Protocol + Remote IP Address + Remote Port Number



TCP Socket (cont.)

```
public class nu_18_socket_server {
  public static void main(String[] args) throws IOException {
    ServerSocket ss = new ServerSocket(50555);
    Socket s = ss.accept();
    System.out.println("connected to Client: " + s.getInetAddress() );

    DataInputStream in = new DataInputStream(s.getInputStream());
    System.out.println(in.readUTF());

    DataOutputStream out = new DataOutputStream(s.getOutputStream());
    out.writeUTF("Thank you from Server ");
    s.close();
  }
}
```



TCP Socket (cont.)

```
public class nu_18_socket_client {
  public static void main(String[] args) throws UnknownHostException, IOException {
    Socket c = new Socket("localhost", 50555);
        /* Data Output Stream / Input Stream from java.io */
    DataOutputStream out = new DataOutputStream(c.getOutputStream());
    out.writeUTF("Hello from " + c.getLocalSocketAddress());

    DataInputStream in = new DataInputStream(c.getInputStream());
    System.out.println("Server Says: " + in.readUTF());

    c.close();
  }
}
```

Important Methods from Socket Object

- Return Remote Port Number `c.getPort()`
- Return Local Port Number `c.getLocalPort()`
- Return Remote IP Address `c.getInetAddress()`
- Return Output Stream `c.getOutputStream()`
- Return Input Stream `c.getInputStream()`

InetAddress

- Get information about who is connecting to the server
- `InetAddress a = c.getInetAddress()`
- `InetAddress a = InetAddress.getByName("nu.edu.pk")`
- `InetAddress[] a = InetAddress.getAllByName("www.google.com")`
- `InetAddress a = InetAddress.getLocalHost()`
- Methods:
  - Get Client Host Name `a.getHostName()`
  - Get Client IP Address `a.getHostAddress()`
  - Is it Local Address? `a.isAnyLocalAddress()`
  - Is it Multicast Address? `a.isMulticastAddress()`
  - Ping Check (ms) `a.isReachable(2000)`

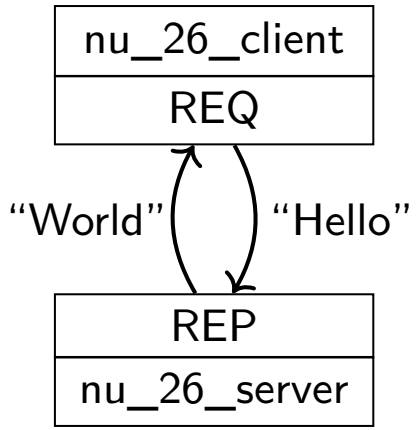
UDP

- `DatagramSocket s = new DatagramSocket(port)`
- `DatagramPacket d = new DatagramPacket(buffer, buffer.length)`
  - `byte[] buffer = new byte[1024]`
- `s.receive(d)`
- `DatagramPacket ack = new DatagramPacket(req.getData(), req.getLength(),`
- `s.send(ack);`

ZeroMQ Overview

- ØMQ (www.zeromq.org)
- MQ Message Queue, Ø Symbolic (zero cost, zero latency, etc.)
- Intelligent Socket Library for Messaging
- FAST (8 Million msg/sec, 30 μ s Latency)
- Platform and Language Independent

Request Reply Message Pattern



- REQ-REP socket pair is in lockstep (Two `s.send()` will give -1 from the `s.recv()` call, and vice versa)
- **Context** helps manage socket (input: # of threads)
- All communication by bytes
- Connection String: inproc for thread to thread, ipc for process to process, tcp for box to box

```
import org.zeromq.ZMQ;

public class nu_26_zmqClient {
  public static void main(String[] args) {
    ZMQ.Context c = ZMQ.context(1);
    ZMQ.Socket  s = c.socket(ZMQ.REQ);
    s.connect ("tcp://localhost:5555");

    s.send ("Hello", 0);
    byte [] msg = s.recv(0);
    System.out.println (new String(msg));
    s.close();
    c.term();
  }
}
```

```
import org.zeromq.ZMQ;

public class nu_26_zmqServer {
  public static void main(String[] args) {
    ZMQ.Context c = ZMQ.context(1);
    ZMQ.Socket  s = c.socket(ZMQ.REP);
    s.bind ("tcp://*:5555");
    while (true) {
      byte [] msg = s.recv (0);
      s.send("World", 0);
    }
    s.close();
    c.term();
  }
}
```

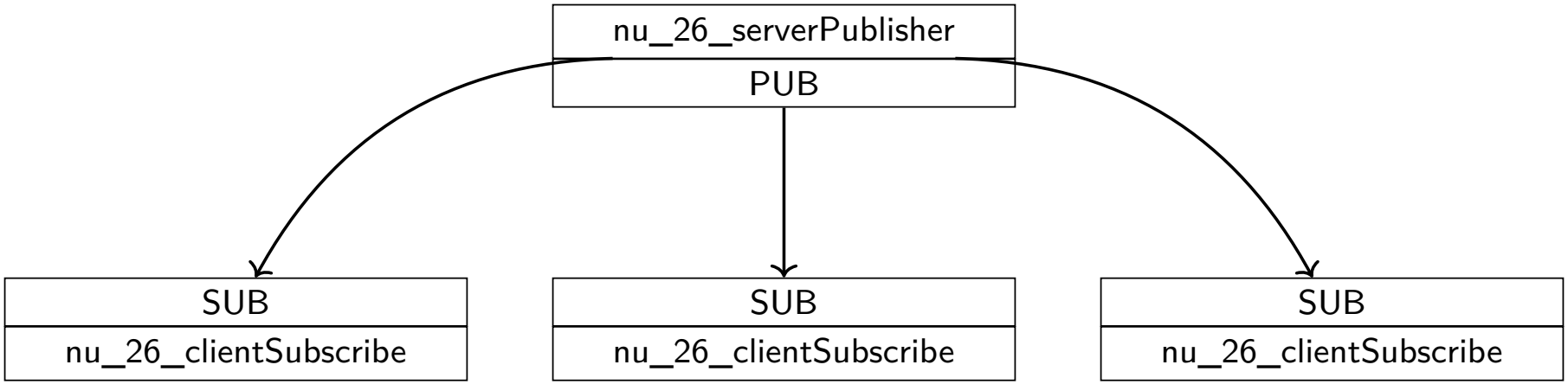
- Compilation & Running

Request Reply Message Pattern (cont.)

```
javac -classpath /usr/local/share/java/zmq.jar nu_26_zmqClient.java
java -Djava.library.path=/usr/local/lib64 -classpath /usr/local/share/java/zmq.jar:. nu_26_zmqClient
```

- Server while loop condition can be set to `!Thread.currentThread().isInterrupted()`

Publisher Subscriber Message Pattern



- Server Pushes Updates to a Number of Clients
- 1 Server Socket linked with Multiple Client Sockets (1:M communication)
- Publisher implements `bind()`, whereas subscribers implement `connect()` methods (it doesn't matter who is server and who is client)
- PUB-SUB socket pair is asynchronous

Publisher Subscriber Message Pattern (cont.)

```
ZMQ.Context c = ZMQ.context(1);
ZMQ.Socket s = c.socket(ZMQ.PUB);
s.bind ("tcp://*:5557");


int count = 0;
while ( /* Condition Check */ ) {
    String str = "Hello World " + count++;
    System.out.println(str);
    s.send(str.getBytes(), 0);
}
s.close();
c.term();
```

```
ZMQ.Context c = ZMQ.context(1);
ZMQ.Socket s = c.socket(ZMQ.SUB);
s.connect ("tcp://localhost:5557");

String filter = "World";
s.subscribe(filter.getBytes());


for (int i = 0; i < 100; i++) {
    byte [] buf = s.recv(0);
    System.out.println (new String(buf));
}
s.close();
c.term();
```

- Synchronization aspects:
  - Subscriber always misses a few messages that the publisher sends
  - Full synchronization if client starts before Server
  - If no subscriber available, then the publisher drops all messages (but they are generated)

Apache Kafka

Overview

- Functions like a distributed messaging system (or a distributed streaming platform)
  - A high throughput, scalable messaging system
  - Distributed, reliable publish-subscribe system
  - Designed as a message queue & Implementation as a distributed log service
  - Lets you process streams of records as they occur
- Originally developed by LinkedIn, now widely popular and adopted by Apache
  - User behavior tracking (site activitities: page views, searches, etc.)
  - Site activity published to central topics, with one topic per activity type.
  - Other examples: payment transactions, geolocation updates, shipping orders, sensor measurements from IoT devices or medical equipment, ...
- Other prominent applicants using Kafka: Netflix, Twitter, Uber, Air BnB, CERN, ...

Kafka Cluster

Working Principals

Basic Architectural View

- Stores streams of **records** (or messages, or events) in categories called **topics** (e.g. folder = topic, file in folder = records)
- Records also known as partitions
- Each data record consists of a key, value, and a timestamp
- Each topic can have zero, one, or many consumers that subscribe to the data

Kafka Cluster (cont.)

Working Principals

Record / Partition Features

- All records are ordered
- All records are replicated. In sync replicas are identified. Replica records are never written into, nor read from.
- Each record has a leader. If a leader dies, its in sync replica is chosen instead.
- Records are immutable
- Records written to page cache of OS, and later written to HDFS or HBase, etc., whichever applicable.

```
# Create Topic
kafka-topics.sh --create --topic example-event --bootstrap-server localhost:9092

# Describe Topic
kafka-topics.sh --describe --topic example-event --bootstrap-server localhost:9092
Topic: example-event    TopicId: xk8JJwuMQS2qb0NEb0rKlw    PartitionCount: 1
                        ReplicationFactor: 1               Configs: segment.bytes=1073741824
Topic: example-event    Partition: 0      Leader: 0    Replicas: 0    Isr: 0
```

Kafka Cluster (cont.)

Working Principals

Reasons for High Throughput

- All writes go to page cache of OS, and later written to disk
- All reads from page cache directly to network socket, through `sendFile()` system call

Physical Components

- **Producer**: Role to send messages to broker with a topic. All new messages are simply appended to a topic.
- **Consumer**: Role to receive messages from broker of a topic of their choice. Topics read on the basis of offset pointer, so fetches may be random.
- **Broker**: One node of a Kafka Cluster (id $= 0 \ldots n$, unique for each node)
- **Zookeeper**: Coordinator of a Kafka Cluster

Kafka Cluster (cont.)

Working Principals

Topic Partitions Features

- Allow the log to scale beyond a size that will fit on a single server.
- Handles an arbitrary amount of data; a topic may have many partitions
- Acts as the unit of parallelism
- Distributed over the servers in the Kafka cluster and each partition is replicated for fault tolerance
- Each partition has one server acts as the *leader* (broker) and zero or more servers act as *followers* (brokers). If the leader fails, one of the followers will automatically become the new leader.

```
kafka-console-producer.sh --topic example-event --bootstrap-server localhost:9092
this is a first message
this is a second message

kafka-console-consumer.sh --topic example-event --from-beginning --bootstrap-server localhost:9
this is a first message
this is a second message
```

Kafka Cluster (cont.)

Working Principals

Message Guarantees

- At Least Once
- At Most Once
- Exactly Once

Operation Modes

- **Point to Point**: Messages persist in a queue, a particular message is consumed by maximum of one consumer only.
- **Publish Scribe**: Messages are persisted in a topic, consumers can subscribe to one or more topics and consume all messages within that topic.

Kafka Cluster (cont.)

Working Principals

Configuration

```
vim /etc/kafka/server.properties
    broker.id = 0 # Must be unique for each broker
    listeners = PLAINTEXT://localhost:9092
    zookeeper.connect = localhost:2181 # All nodes in the cluster

kafka-server-start.sh /etc/kafka/server.properties
```

Additional Non-Examinable Topics

Record / Partition Compression

- Most recent value of key is retained in page cache.
- Solved Example (Discussion)

Zero Copy Concept

- Data Directly written to NIC through sendfile syscall, leading to zero copy overhead to application buffers.

Kafka Connect

- Transmit data from relational databases or dataframes
- Allows to continuously ingest data from external systems into Kafka (and vice versa)
- **Connectors** implement custom logic for interacting with an external system.

Additional Non-Examinable Topics (cont.)

File ↔ Kafka connector

```
vim /etc/kafka/connect-standalone.properties
  plugin.path=/usr/share/java/kafka/connect-file-3.2.0.jar

vim /etc/kafka/connect-file-source.properties
  connector.class=FileStreamSource
  file=test.txt
  topic=connect-test

vim /etc/kafka/connect-file-sink.properties
  connector.class=FileStreamSink
  file=test.sink.txt
  topics=connect-test

connect-standalone.sh connect-standalone.properties
          connect-file-source.properties connect-file-sink.properties

kafka-console-consumer.sh --bootstrap-server localhost:9092
          --topic connect-test --from-beginning
{"schema":{"type":"string","optional":false},"payload":"foo"}
{"schema":{"type":"string","optional":false},"payload":"bar"}
```

Overview

- Spark libraries for distributed machine learning
- org.apache.spark.mllib
  org.apache.spark.ml
- mllib: Original API built on top of RDD's
- ml: mllib Migration path with Higher level API with RDD DataFrames for constructing ML Pipelines

Overview (cont.)

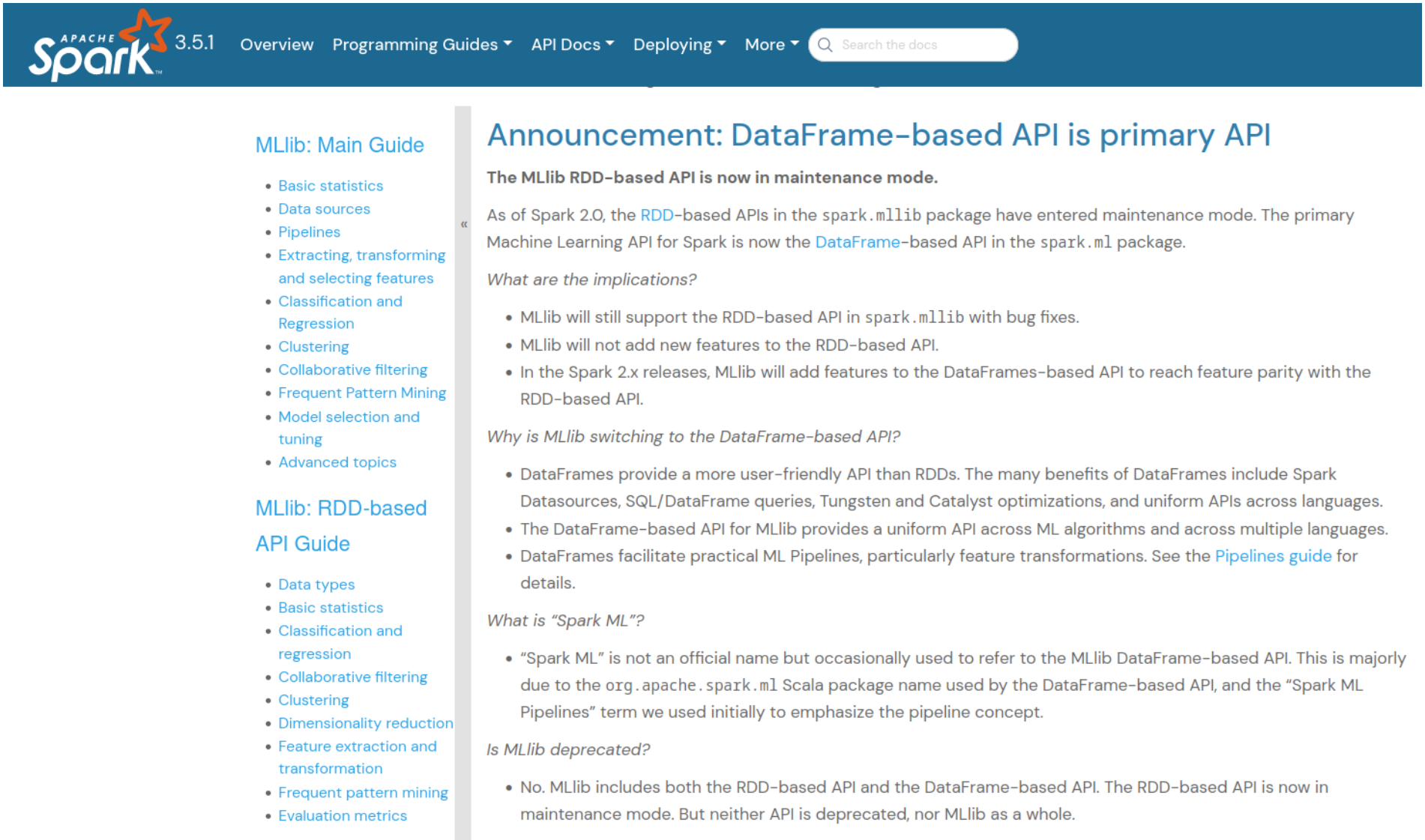


Figure 18: Grab: https://spark.apache.org/docs/latest/ml-guide.html

Spark Data Frames vs Pandas Data Frames

Python DF

```
import pandas as pd
pdf  = pd.read_csv('data.csv')
```

Spark DF

```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setMaster("local").setAppName("My App")
  sc = SparkContext (conf = conf)  # If New session
# sc = SparkContext.getOrCreate(); # If already created elsewhere

 sdf = spark.read.csv('training.csv', header=True)
```

	Spark DF	Pandas DF
Operations in Parallel	Yes	Not out of the box
Lazy Evaluation	Yes	No
Immutable	Yes	No
Functional Programming	Yes*	Yes*

* But mutability makes it partially similar

Spark Data Frames vs Pandas Data Frames (cont.)

Parallel

```
 df_grouped = pdf.groupby("category").count() # Runs in Single Thread
sdf_grouped = sdf.groupBy("category").count() # Runs in Parallel on Cluster
```

Lazy Evaluation

```
pdf_filtered = pdf[pdf["price"] > 100]         # Executes immediately

sdf_filtered = sdf.filter(sdf["price"] > 100) # Not Executed
sdf_filtered.show()                            # Executes Now
```

Binary Images

```
from pyspark.sql.types import BinaryType
spark.sql("set spark.sql.files.ignoreCorruptFiles=true")

df = spark.read.format("binaryFile").option("pathGlobFilter", "*.jpg")
                                    .option("recursiveFileLookup", "true")
                                    .load(file_path)
```

```
DataFrame[path: string, modificationTime: timestamp, length: bigint, content: binary]
```

Scikit-learn vs MLLib

- Scikit-Learn: ML Library leveraging Numpy, Scipy, Matplotlib (Non-Distributed, works on mutable datasets)

	Scikit-Learn	MLLib
Datasets	Mutable (Inplace Column Update)	Immutable (New Columns)
Scalability	Single Machine	Distributed Machines
Model Deployment	ML Flow	ML Flow

MLLib Data Types (Vectors and Matrices)

- Local vs Distributed
- Sparse vs Dense

MLLib Dense Vectors/Matrices

```
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.linalg import Matrices

# Local Vectors (Dense, Sparse)
v1 = Vectors.dense([1.1,-2,0.3,45,-3.5])
print(v1)                    # [1.1,-2.0,0.3,45.0,-3.5]
m1 = Matrices.dense(3,2,[1,2,3,4,5,6])
print(m1)                    # Column Major: 1, 4 \\ 2, 5 \\ 3, 6
```

- Note: Dense vectors/matrices of Numpy / python list also supported

```
import numpy as np

dv1 = np.array([1.0, 0.0, 3.0])
dv2 = [1.0, 0.0, 3.0]
```

MLLib Data Types (Vectors and Matrices) (cont.)

Sparse Vectors/Matrices

```
v2 = Vectors.sparse(5, [1,3], [10,20])
print(v2)                    # (5,[1,3],[10.0,20.0])
v3 = Vectors.sparse(5, [0, 1, 2], [10, 20,30])
print(v3)                    # (5,[0,1,2],[10.0,20.0,30.0])
m2 = Matrices.sparse(3, 3, [0, 1, 2, 3], [0, 0, 2], [9, 6, 8])
print(m2)                    # Compressed Sparse Column Major: 9, 6, 0 \\ 0, 0, 0\\0, 0, 8
```

- Label + Label Feature Vector (Dense/Sparse)

```
from pyspark.mllib.regression import LabeledPoint

p1 = LabeledPoint(1.0, [1.0, 0.0, 3.0])
print(p1)                    # (1.0,[1.0,0.0,3.0])

p2 = LabeledPoint(0.0, SparseVector(3, [0, 2], [1.0, 3.0]))
print(p2)                    # (0.0, (3,[0,2],[1.0,3.0]))
```

MLLib Data Types (Vectors and Matrices) (cont.)

Distributed Matrices

RowMatrix, IndexedRowMatrix, CoordinateMatrix, BlockMatrix

- RowMatrix -> Local Vector Row -> Multiple Partition Storage

```
from pyspark.mllib.linalg.distributed import RowMatrix

rows = sc.parallelize([[1,2,3], [4,5,6], [7,8,9], [10,11,12]])
row_matrix = RowMatrix(rows)
```

- IndexedRowMatrix -> RowMatrix + Meaningful Row Indices

```
from pyspark.mllib.linalg.distributed import IndexedRow, IndexedRowMatrix

from pyspark.mllib.linalg.distributed import IndexedRow, IndexedRowMatrix
indexed_rows = sc.parallelize([
    IndexedRow(0, [0,1,2]),
    IndexedRow(1, [3,4,5]),
    IndexedRow(2, [6,7,8])
])
indexed_rows_matrix = IndexedRowMatrix(indexed_rows)
```

MLLib Data Types (Vectors and Matrices) (cont.)

- Coordinate Matrix -> (i, j, v) format

```
from pyspark.mllib.linalg.distributed import CoordinateMatrix, MatrixEntry

matrix_entries = sc.parallelize([MatrixEntry(0, 5, 3),
                                 MatrixEntry(1, 2, 6),
                                 MatrixEntry(0, 0, 9)])
c_matrix = CoordinateMatrix(matrix_entries)
```

- Block Matrix -> Nested Matrix Entries (Matrix in Matrix)

```
from pyspark.mllib.linalg import Matrices
from pyspark.mllib.linalg.distributed import BlockMatrix

blocks = sc.parallelize(
[((0, 0), Matrices.dense(3, 3, [1, 2, 1, 2, 1, 2, 1, 2, 1])),
 ((1, 1), Matrices.dense(3, 3, [3, 4, 5, 3, 4, 5, 3, 4, 5])),
 ((2, 0), Matrices.dense(3, 3, [1, 1, 1, 1, 1, 1, 1, 1, 1]))])

b_matrix = BlockMatrix(blocks, 3, 3)
local_mat = b_matrix.toLocalMatrix().toArray()
```

ML Datatypes

- ```
  from pyspark.mllib.linalg import Vectors
  from pyspark.ml.linalg    import Vectors
  ```
- ```
  from pyspark.mllib.linalg import Matrices
  from pyspark.ml.linalg    import Matrices
  ```

Machine Learning Pipeline

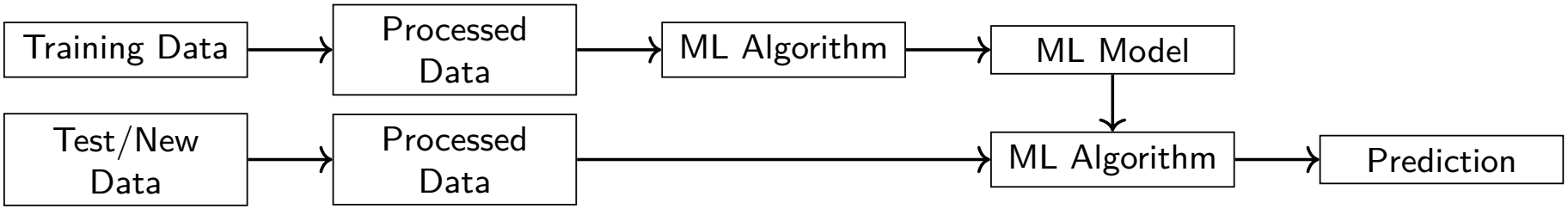


Figure 19: Typical Process Flow (Supervised Learning)

Pipeline Components

DataFrame, Transformer, Estimator, Parameters

API Call: `pyspark.ml.feature`
Pipeline: Ordered Sequence of Transformers/Estimators as stages

Transformer

- `.transform()`
- Convert one dataframe to another

Machine Learning Pipeline (cont.)

- Text Data Transformers
- Numeric Transformations (Binarizer, Bucketizer, MaxAbsScalar, MinMaxScalar, Normalizer, QuantileDiscretizer, ...)
- Additional Transformations (Discrete Cosine Transform, ElementwiseProduct, Imputer, Interaction, ...)

Estimator

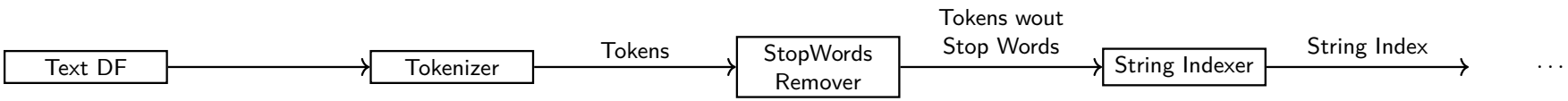
- A learning activity that fits or trains on provided data
- Convert Dataframe to a model (transformer)
- `.fit()`

Parameters

- Aids in Transforming/Estimating
- Key value pairs, E.g. maxIter=10 for setting maximum iterations

Machine Learning Pipeline (cont.)

```
sentence_data_frame = spark.createDataFrame([
(0, "Hi I think pyspark is cool ","happy"),
(1, "All I want is a pyspark cluster","indifferent"),
(2, "Yes, I can","happy")
], ["id", "sentence", "sentiment"])

from pyspark.ml.feature import Tokenizer
tokenizer = Tokenizer(inputCol="sentence", outputCol="words")
tokenized = tokenizer.transform(sentence_data_frame)

from pyspark.ml.feature import StopWordsRemover
remover = StopWordsRemover(inputCol="words", outputCol="meaningful_words")
meaningful_data_frame = remover.transform(tokenized)
meaningful_data_frame.select("words","meaningful_words")

from pyspark.ml.feature import StringIndexer
indexer = StringIndexer(inputCol="sentiment", outputCol="categoryIndex")
indexed = indexer.fit(meaningful_data_frame).transform(meaningful_data_frame)

from pyspark.ml.feature import HashingTF
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
hashed    = hashingTF.transform(indexed)
```



Machine Learning Pipeline (cont.)

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer

training = spark.createDataFrame([
    (0, "a b c d e spark", 1.0),
    (1, "b d", 0.0),
    (2, "spark f g h", 1.0),
    (3, "hadoop mapreduce", 0.0)
], ["id", "text", "label"])

tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.001)

pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

# Fit the pipeline to training documents.
model = pipeline.fit(training)
```

Custom

```
import pyspark.sql.functions as F
from pyspark import keyword_only
from sparkml_base_classes import TransformerBaseClass

class CustomTransformer(TransformerBaseClass):

    @keyword_only
    def __init__(self, column_name=None, value=None):
        super().__init__()

    def _transform(self, ddf):
        # Code here

class customEstimator(EstimatorBaseClass):

    @keyword_only
    def __init__(self, column_name=None, value=None):
        super().__init__()

    def _fit(self, ddf):
        # Code here

customT = CustomTransformer()
customE = CustomEstimator()
my_pipeline = Pipeline(stages=[customT, customE])
```

Splitting Datasets

```
data = sc.parallelize([1,2,3,4,5,6,7,8,9,10])
training, test = data.randomSplit([0.6, 0.4])
training.collect()
[3, 4, 5, 6, 7, 9]
test.collect()
[1, 2, 8, 10]
```

Supported Algorithms