EE3801 Cheatsheet
Intro to Data Engineering
github.com/securespider

01.1 Intro
Data science vs engineering
- **Science** - Learn, optimise, analytics, aggregate and labelling
- **Engineering** - Cleaning, data storage, logging, sensors, pipelines

Data structure
Unstructured data
- Chaotic no order to data

Structured data
- Data stored access in the same format

Semi structured data
- Can contain both forms of data
- Some structure but not all data points follow same format

Big data
Volume, Variety, Variability

Velocity High rate of data generation
- Must create a robust and scalable pipeline

Raw Data
- Tend to have gaps

Data wrangling
Used to understand raw data

Discovery Understand what is in your data

Structure

Cleaning Dealing with gaps (nulls), outliers, formatting bugs

Enrichment Derive other data from other information/ additional data augmentation (feature selection)

Validation Verify data quality, sources

Publishing Give data scientist

Process
Extraction Retrieve raw data from unstructured pool and migrate to temp repo

Transformation Structure enrich and convert raw data

Loading Loading structured data into data warehouse

Data warehouse
Decision support system storing historical data from organisations

Data Pipeline
- Processing underlying raw data in ordered sequence of steps

01.2 Data Pipelines
Considerations
Big data
Velocity Streaming, captured and processed in real time

Volume Scalable wrt time

Variety Recognise and process diff formats

Business
- Handling streaming data?
- How much data to expect (Time horizon/how much storage consumed)
- What type/how much processing in DP
- Where is data source? Need micro-services?

Architecture
Batch-based DP
- Analysis of data that has been stored over a period of time
- N independent tasks to process with k stages
- Each stage takes max of T time process input
- Diff stage can operate concurrently
- $t(N,k) = T \times (N + k - 1)$

Streaming-based DP
- Processing as data flows through system
- Logging and persistent result storage

Lambda Architecture
- Combination of batch and streaming
- Separate processing engine for "batch" and "speed" layers combining in "service" layer
- Accounts for real-time streaming and historical batch analysis
- Encourage raw data storage and create new dst for queries
- Min errors for both layers reliably at fast speeds

Kappa Architecture
- Replay data and process both layers in same single stream processing engine
- Good for big data architecture with cheaper hardware and focus on stream

Design
1. Identify application and decide if DP needed
2. Identify DP category (architecture)
3. Understand working mechanism, parameters/variables

04. Big Data Computing Technology Platform
Packaging
Compact Nodes closely packaged in racks where nodes are not attached to peripherals

Slack Nodes attached to peripherals connected remotely

Interconnection Medium
Considerations
1. Available link speeds
2. Message Passing Interface (MPI) latency
3. Network processor/routing mechanism/flow control
4. Differing network topologies

Self routing/Destination tag
- Every processor can be routed to every memory without external controller
- Switch should know what stage it is in to know which bit to look for
- Bit of stage defines which output interface it leaves (0-above, 1-below)

Control
Centralized Nodes owned, ctrl by central operator
- Easy to manage
- Used by compact and slack clusters

Decentralized Nodes have individual owners
- Minimize coupling and can be used w many OS
- Only slack can have

Homogeneity
Homogeneous

Heterogeneous

Resource sharing
Share-nothing Each node do itself and send results together after

Shared-disk When one node fail the other take over
- Fault tolerance via checkpoints

Shared-memory Connected via SScalable Coherence Interface ring
- All common data/instruction written in shared space