Summer 2019 Software Engineering Internship at Symantec Corporation

Table of Contents

Company & Team Overview

About Symantec

Symantec Corporation (NASDAQ: SYMC), the world's leading cyber security company, allows organizations, governments, and people to secure their most important data wherever it lives. Its Integrated Cyber Defense (ICD) Platform unifies products, services, and partners, protecting enterprises against sophisticated threats across endpoints, networks, applications, and clouds.

About Data Loss Prevention (DLP) Product – ServerApps Team

I was part of the DLP Product - ServerApps Team, directed by Pushkar Tiwari. The DLP product is designed to prevent data loss across four channels at the enterprise level.

- Endpoint (Printers, USB, etc.)
- Network (Outgoing Internet traffic, ICAP, intranet traffic, etc.)
- Databases (Box cloud repositories, cloud databases, etc.)
- Email (SMTP, MTA, etc.)

The ServerApps team managed the cloud DLP product offering, This meant monitoring the client's email, network traffic, cloud databases, etc. through the SYMC cloud.

Background Concepts & Motivation

Background – Detectors & Partitions

The DLP backend is structured into detectors and partitions. Detectors (client instances of DLP functionality) may be thought of as fruits/items of varying size. These are to be fit into partitions (Amazon EC2 instances) which are boxes/bins with a constant (finite) capacity. Refer to Appendix – Terminology for fuller explanation.

Business Value

Symantec realizes that Amazon EC2 instances represent a significant operating cost for the business. Thus, there are clear monetary incentives to minimize the total number of partitions that are used, by packing detectors into partitions as densely as feasible.

Optimization Formulation

Problem Statement: Assign finite number of detectors, to partitions such that total number of partitions used is minimized subject to some application-specific constraints

Optimization Objective: Minimize number of partitions used. The partitions also need to be balanced.

Decision Variables: Assignments of detectors (items) to partitions (bins)

Key Constraints:

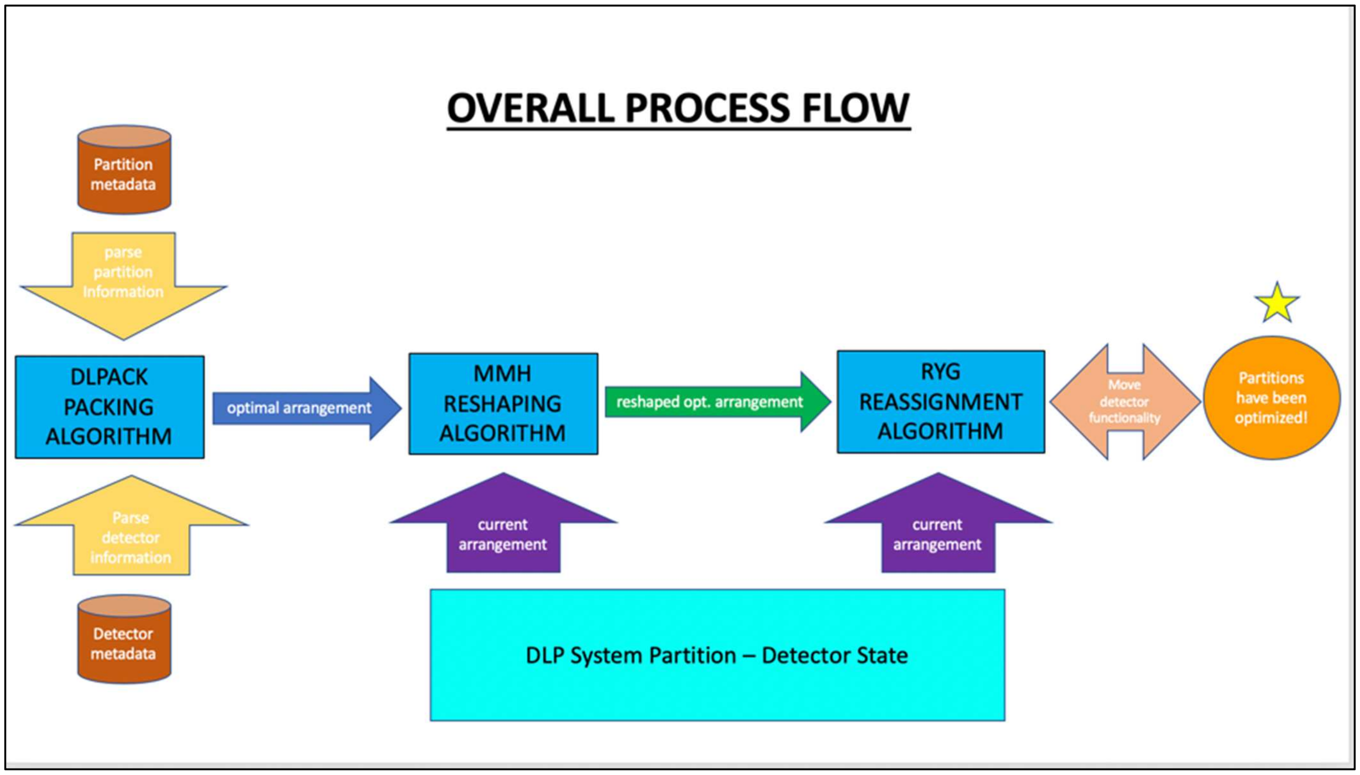
- *Partition Memory Upper-bound*: No more than Y GB of aggregated profile data can be stored on a partition 15GB
- *Number of Detectors Upper Bound*: No more than Z detectors may be stored on a given partition instance (80)
- *Type of traffic*:
  - DIM: Data in motion has specific latency requirements. In practice, this means that DIM detectors are to be allocated to their partitions (i.e. no mixing).
  - DAR: Has no latency requirements
- *Dedicated partition*: If profile size exceeds X GB, then the partition must be dedicated to that task. (1.2GB)

Problem Solving Approach

A well-read reader may notice that this is a variation of the classic bin-packing problem studied in combinatorial optimization. This is an NP hard problem. So, finding an exactly optimal solution is difficult, so a heuristic is oftentimes more accessible.

I began with an Exploratory Data Analysis (EDA) of production data provided to me by Vincent Ploccienik, a software engineer.

I was able to design the following system to solve the problem using incremental steps:

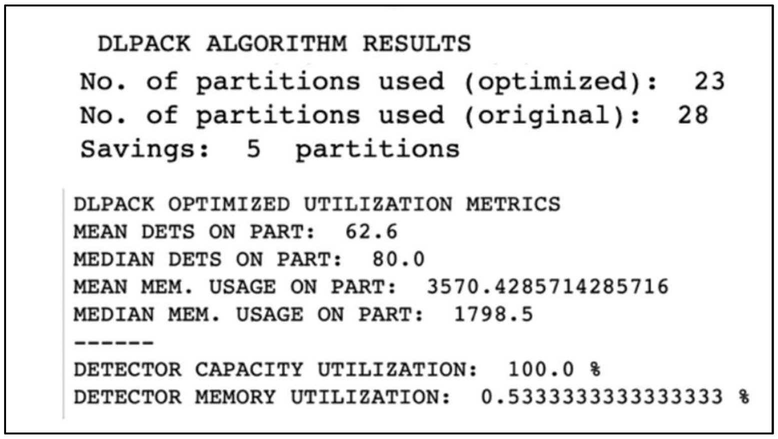

OVERALL PROCESS FLOW
Partition metadata
parse partition information
DLPACK PACKING ALGORITHM
Parse detector information
Detector metadata
optimal arrangement
MMH RESHAPING ALGORITHM
reshaped opt. arrangement
RYG REASSIGNMENT ALGORITHM
current arrangement
current arrangement
DLP System Partition – Detector State
Move detector functionality
Partitions have been optimized!

Each of the above blue boxes represents an algorithm I devised:

- **DLPack:** A tailor-made *packing algorithm* to define a target optimal solution
- **MMH:** A *reshaping algorithm* to make the optimal solution resemble the current allocation w/o sacrificing optimality (reduces the # of moves engineers have to make later)
- **RYG:** A *reassignment algorithm* to actually move the detectors from one partition to another to arrive at the optimal state

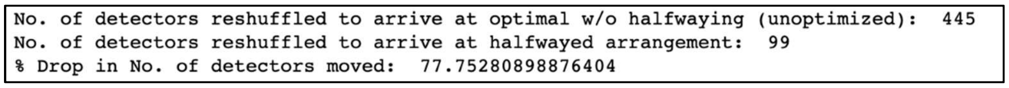
Key Results

On production data, each algorithm provided the following positive results:



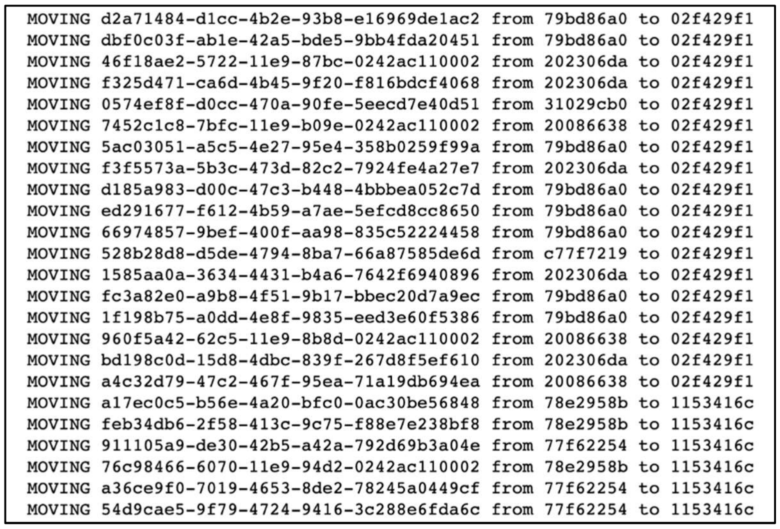
```
DLPACK ALGORITHM RESULTS
No. of partitions used (optimized):  23
No. of partitions used (original):  28
Savings:  5  partitions

DLPACK OPTIMIZED UTILIZATION METRICS
MEAN DETS ON PART:   62.6
MEDIAN DETS ON PART:   80.0
MEAN MEM. USAGE ON PART:   3570.4285714285716
MEDIAN MEM. USAGE ON PART:   1798.5
------
DETECTOR CAPACITY UTILIZATION:   100.0 %
DETECTOR MEMORY UTILIZATION:   0.5333333333333333 %
```

MMH reduced the number of moves required to arrive at optimality from current suboptimal state by 77.752%. Without MMH, 445 out of 641 detectors (i.e. ~70%) would have to be reshuffled. With MMH, only about 99 out of 641 detectors (i.e. ~15%) need to be moved to arrive at optimality without sacrificing the total number of bins used – essentially a free lunch. This makes the solution more implementable by engineers, and also reduces the time + effort required to move from suboptimal to optimal states.



```
No. of detectors reshuffled to arrive at optimal w/o halfwaying (unoptimized):  445
No. of detectors reshuffled to arrive at halfwayed arrangement:  99
% Drop in No. of detectors moved:  77.75280898876404
```

Finally, the RYG algorithm uses the output of the MMH algorithm to specify a move list for future engineers (or a computer program) to be able to make the necessary changes. For example, the final move list may look like:



```
MOVING d2a71484-d1cc-4b2e-93b8-e16969de1ac2 from 79bd86a0 to 02f429f1
MOVING dbf0c03f-ab1e-42a5-bde5-9bb4fda20451 from 79bd86a0 to 02f429f1
MOVING 46f18ae2-5722-11e9-87bc-0242ac110002 from 202306da to 02f429f1
MOVING f325d471-ca6d-4b45-9f20-f816bdcf4068 from 202306da to 02f429f1
MOVING 0574ef8f-d0cc-470a-90fe-5eecd7e40d51 from 31029cb0 to 02f429f1
MOVING 7452c1c8-7bfc-11e9-b09e-0242ac110002 from 20086638 to 02f429f1
MOVING 5ac03051-a5c5-4e27-95e4-358b0259f99a from 79bd86a0 to 02f429f1
MOVING f3f5573a-5b3c-473d-82c2-7924fe4a27e7 from 202306da to 02f429f1
MOVING d185a983-d00c-47c3-b448-4bbbea052c7d from 79bd86a0 to 02f429f1
MOVING ed291677-f612-4b59-a7ae-5efcd8cc8650 from 79bd86a0 to 02f429f1
MOVING 66974857-9bef-400f-aa98-835c52224458 from 79bd86a0 to 02f429f1
MOVING 528b28d8-d5de-4794-8ba7-66a87585de6d from c77f7219 to 02f429f1
MOVING 1585aa0a-3634-4431-b4a6-7642f6940896 from 202306da to 02f429f1
MOVING fc3a82e0-a9b8-4f51-9b17-bbec20d7a9ec from 79bd86a0 to 02f429f1
MOVING 1f198b75-a0dd-4e8f-9835-eed3e60f5386 from 79bd86a0 to 02f429f1
MOVING 960f5a42-62c5-11e9-8b8d-0242ac110002 from 20086638 to 02f429f1
MOVING bd198c0d-15d8-4dbc-839f-267d8f5ef610 from 202306da to 02f429f1
MOVING a4c32d79-47c2-467f-95ea-71a19db694ea from 20086638 to 02f429f1
MOVING a17ec0c5-b56e-4a20-bfc0-0ac30be56848 from 78e2958b to 1153416c
MOVING feb34db6-2f58-413c-9c75-f88e7e238bf8 from 78e2958b to 1153416c
MOVING 911105a9-de30-42b5-a42a-792d69b3a04e from 77f62254 to 1153416c
MOVING 76c98466-6070-11e9-94d2-0242ac110002 from 78e2958b to 1153416c
MOVING a36ce9f0-7019-4653-8de2-78245a0449cf from 77f62254 to 1153416c
MOVING 54d9cae5-9f79-4724-9416-3c288e6fda6c from 77f62254 to 1153416c
```

Conclusion & Further Steps

My direct managers, Daniel Kim & Vincent Ploccienik, as well as Pushkar Tiwari (Director of Development) were convinced of the business value of my work. They are making plans to integrate it with their present detector-partition allocation framework, using GRPC (Google Remote Procedural Calling) interface, designed by another engineer. I also contributed to Symantec's knowledge base, by documenting my methodology & results on Confluence – the company's employee knowledge sharing Wiki portal. Based on my excellent work, I have been given the opportunity to continue working with the team, on a part-time remote basis.

Appendix

I. Technical Terminology

Detector

A detector is an abstraction that represents a single customer's detection functionality to be monitored, as well as the associated set of profiles that are to be checked against. For example, a detector could represent a customer's email traffic (SMTP/IMAP) that is to be monitored, while the profiles to check against could be SSNs of the format AAA-GG-SSSS. Another example would be monitoring outgoing internet traffic (ICAP) from employees' browsers that is to be checked for sensitive data flows before the leave the client's network into the open Internet. Alternatively, a detector may represent a passive scan of a customer's cloud repository, where the profile to identify are perhaps sensitive health records, or phone numbers, etc. The first two are examples of data-in-motion (DIM), while the latter is an example of data-at-rest (DAR). Of course, since the DLP product is placed inline with the customer's data flow, DIM has specific latency requirements (the lower the better), while DAR has less stringent latency requirements, due to it being simply a passive scan conducted at specified intervals. Thus, a detector has the following attributes:

- DetectorID (unique identifier for each detector)
- Version (either V2, V3)
- Profile Size (some value in MB)
- DIM/DAR (what type of data does it represent)

Partitions

A partition is simply a formalism of Amazon EC2 (Elastic Cloud Compute) instance that is responsible for carrying out the processing of a single (or multiple detectors). It has finite memory capacity, a version, and can be either dedicated or shared. This means that it may contain either exactly one detector ('dedicated') or multiple detectors ('shared'). This leads to the following attributes:

- PartitionID (unique identifier for each partition)
- Version (either V2, V3)
- Dedicated/Shared
- Maximum Total Memory Limit
- Maximum Number of Detectors

II. Algorithms

DLPack – A Packing Algorithm

- Phase 0 (Setup)
    - Separate detectors based on V2, V3 and then route to appropriate set of partitions
    - Classify each detector as XL, Regular (R), XS according to Profile Size such that:
        - >= 1.2 GB => XL
        - > 0, less than 1200MB => R
        - = 0 => XS
- Phase 1 (XL Mapping)
    - Put in all XL detectors into dedicated partitions
- Phase 2 (R Mapping)
    - Sort remaining detectors (R) in decreasing order of profile size
    - Try putting each detector in partition beginning with the first one
        - Open a new partition if last partition is FULL
- Phase 3 (XS Mapping)
    - Compute the packing ratio for each partition as "λ_{DLPack} = # of free detector space remaining ÷ amt. of free memory space remaining".
    - Start putting XS detectors in order of descending packing ratio. So, detectors are to be packed into partition with maximal packing ratio, and then move onto next partition, and so on.
    - Return # of partitions, det-par mapping

(*) Partition is FULL if:

(1) Total profile size limit has been reached

OR

(2) Total # of detectors limit has been reached

Meet-me-halfway (MMH) – A Reshaping Algorithm

- For each partition in optimal arrangement that is SHARED:
    - Identify all the XS detectors. Call it xs_list.
    - Compare to corresponding partition in current arrangement to determine which XS detectors are of the *equivalent type* (i.e. dim/dar and v2/v3). Call it 'xs equivalent list'.
    - Remove each detector in xs_list to holding zone
    - For each detector in xs_equivalent_list:
        - Search for its location in the optimal arrangement + holding zone
        - Move it from its previous location in the optimal arrangement/holding zone to current partition in optimal arrangement
- Return the *reshaped optimal arrangement*

RYG (Red – Yellow – Green) – A Reassignment Algorithm

- All partitions are initially RED.
- For each partition, mark YELLOW and do the following:
    - Use the optimal allocation to identify detectors that need to be:
        - Kept in place
        - Added to the yellow partition
    - Remove all other detectors in the yellow partition to holding zone
    - Search Red buckets + HZ for each detector to be added to yellow partition
    - Remove those detectors from their previous location, and add to yellow partition.
    - Mark the current partition as GREEN.
- End when all partitions are GREEN.