CS236635
Network Functions Virtualization (NFV)

Class 10

Danny Raz
Last week

• Micro services ..... 

Before that:

• NFV - moving NF to the cloud

This week

• Resource allocation
• Performance
• Placement
Resource allocation in the Cloud

- Where to acquire resources (CPU, Storage)?
  - building the next data center (Google)
  - getting EC2 resources and how much (smaller users)

- Where to place the service and the data?
  - VM placement
  - service/data migration

- Which location should serve a specific request?
  - Load balancing

Key feature: cost effectiveness ➔ resource management
Resource allocation in the Cloud

Key feature: cost effectiveness → resource management

• Where to acquire resources (CPU, Storage)?
  – building the next data center (Google)
  – getting EC2 resources and how much (smaller users)

• Where to place the service and the data?
  – VM placement
  – service/data migration

• Which location should serve a specific request?
  – Load balancing

focus on multidimensionality
This part

- Bin packing
  - Dual bin packing
  - Dual vector bin packing

- Online algorithms
  - Lower bounds
  - Placement algorithm

- Performance evaluation
  - Data sets
  - Results
Bin Packing and dual Bin Packing

- **Bin Packing**: What is the minimal number of bins required to pack all given packages?
- **Dual Bin Packing**: what is the maximum number of packages that can be packed in a set of given bins?
Multi-dimensional Bin-Packing

- Each physical resource (host, switch) is a bin with multi-dimensional capacities
- Each request is a collection of multi-dimensional elements
- Need to fill bins with elements w/o breaching capacities
Dual vector Bin-Packing

- What is the maximum number of packages that can be packed in a set of given bins?
  - Each bin and each package may have several dimensions
  - Much more complicated to solve
Online algorithms

- In order to analyze realistic scenarios where items (VM requests) arrive over time we consider the online variant of the VM placement problem where requests arrive dynamically – one at a time.
- This corresponds to the online variant of dual bin-packing.
- Typically, an online algorithm is analyzed by its performance, compared to the optimal solution, on a worst-case input.
- The competitive ratio of Alg:

\[ R(Alg) = \lim_{N \to \infty} \left( \sup_{\sigma: Opt(\sigma) \geq N} \frac{Alg(\sigma)}{Opt(\sigma)} \right) \]
Optimization - Placement Goals

Focus on this for now

Which to use?

Maximize Revenue

Minimize energy consumption

Minimize customer rejection rate

Service robustness
Dual vector Bin-Packing

Given:
- A sequence of \( n \) \( d \)-dimensional vectors \( p_1, \ldots, p_n \)
- \( m \) \( d \)-dimensional bins \( b_1, \ldots, b_m \)

Find:
- A subset of the vectors, and a partitioning of this subset \( A_1, \ldots, A_m \) of this set into the \( m \) bins

Such that:
- In every dimension, \( \bar{A}_i \) is at most \( b_i \), where \( \bar{A}_i \) is the vector sum of items in \( A_i \), i.e. \( \bar{A}_i = \sum_{j \in A_i} p_j \)

The goal is to maximize the size of the selected subset
Any fair online algorithm has a competitive ratio of at least 1/2

Any fair online deterministic algorithm has a competitive ratio of at most 2/3

Any fair randomized online algorithm has a competitive ratio of at most 4/5

The competitive ratio of worst-fit (leave the most empty space) is exactly 1/2

The competitive ratio of best-fit (leave the least empty space) is $\frac{n}{2n-1}$
Multidimensional is hard

- For $d > 1$, there is no generic lower bound on the competitive ratio.

- In fact, a much stronger claim holds:
  - For $d > 1$ dimensions, *every* deterministic fair online $d$-dimensional dual vector bin-packing algorithm has 0 competitive ratio.
Random order ratio

- Changing the metrics - Compared to the optimal solution, on a random permutation of the worst-case input

- The random order ratio of Alg:

\[ RC(Alg) = \lim_{N \to \infty} \left( \sup_{\sigma:Opt(\sigma) \geq N} \frac{E_L(Alg(\sigma_L))}{Opt(\sigma)} \right) \]

- Does Not help 😞

- Claim: For d>1 dimensions, there exists a fair online d-dimensional dual vector bin-packing algorithm with a diminishing random order ratio
Random order ratio

Proof Sketch:

- [...] For any $\varepsilon$, expected number of placed elements can be made $< \varepsilon$

- Claim: For $d>1$ dimensions, there exists a fair online $d$-dimensional dual vector bin-packing algorithm with a diminishing random order ratio
For two identical bins and two dimensions, there exists a random online dual vector bin-packing algorithm with competitive ratio of $\frac{1}{4}$ for accommodating inputs.

**Algorithm:**
- Randomly pick a dimension, $D$, and mark both bins as “open”
- For every item that arrives:
  - If its size in $D$ is $\leq$ the size in the other dimension, discard
  - If first bin is “open” and fits the item, place in first bin
  - If first bin is “closed”, place in second bin
  - If first bin is “open” and does not fit the item, close the bin and discard the item

**Correctness:**
- We are always placing items that are large in the selected dimension they always fit in the other
- At least half of the items are no smaller in size in one of the dimensions
- If we happen to choose the right dimension (probability $\frac{1}{2}$), we will fit at least $\frac{1}{2}$ of the items - The algorithm will fit $\frac{1}{2} \times n \times \frac{1}{2} - 1$
Main theoretical results

- For two identical bins and two dimensions, there exists a random online dual vector bin-packing algorithm with competitive ratio of $\frac{1}{4}$ for accommodating inputs.

- For two identical bins and two dimensions, and for accommodating inputs, there exists an online dual vector bin-packing algorithm that achieves $\frac{7}{16}$ approximation in linear time.

- There exists an online dual vector bin-packing algorithm with 8-lookahead that achieves $\frac{7}{16}$ competitive ratio for two identical bins in two dimensions.

- For $d$ dimensions and $k$ bins, with $kd$ look ahead one can fit $\frac{1}{kd}$ items in the first bin.

- For $k$ identical bins and $d$ dimensions, there exists a random online $d$-dimensional dual vector bin-packing algorithm with competitive ratio of $\frac{1}{d^2} - k$.

Raz, Segall, Goldstein “Multidimensional Resource Allocation in Practice” Systor, 2017
Practical heuristics

For clarity, presented for $d=2$:

- **OBP** (Orientation-Based Placement) – try to place tall items in tall bins and wide items in wide bins
- **DistFromDiag** – Choose a bin such that distance of the corner of the new item from diagonal of the bin would be smallest
- **BUP** (Balanced Utilization Placement) – try to balance the overall utilized resources between the bins (extension of Worst Fit to $d$-dimensions)
- **Combined** – OBP + BUP as tie-breaker
- **No silver bullet !**

![Diagram](image)
Performance evaluation

Google Data

Percentage of Accommodated Requests vs. Percentage of Allocated Bins for different methods:
- Random
- DistFromDiag
- OBP
- BUP
- Combined
- Adaptive
Performance evaluation

Nokia Data

Percentage of Accomodated Requests

Percentage of Allocated Bins

- Random
- DistFromDiag
- OBP
- BUP
- Combined
- Adaptive
Performance evaluation

Amazon Data

![Graph showing the percentage of accommodated requests against the percentage of allocated bins for different methods: Random, DistFromDiag, OBP, BUP, Combined, and Adaptive. The graph illustrates the performance of these methods with varying allocation percentages.]
To REMEMBER

- Real placement problems in the cloud are **multidimensional**
- **Inherently** different from one dimension (and much harder)
- **Random order** ratio
- No silver bullet – it’s all about the **data**
Is this useful for NVF

- **Placement** = allocating physical resources given customer demand
  - a.k.a. virtual datacenter (VDC) embedding
Placement in OpenStack

- Reminder
  - OpenStack: open-source project for managing your own Cloud
  - Nova: OpenStack project that deals with Compute (VM) management
- OpenStack deals with a single DC
Placement in **OpenStack**

- OpenStack: open-source project for managing your own Cloud deals with a single data center
- Nova: OpenStack project that deals with Compute (VM) management

![Diagram showing host selection process](image)
Placement in OpenStack

- Comes with single built-in optimization parameter – **utilized RAM**
  - i.e., the score for each location is the utilized RAM (%) after placement
- Supports two policies
  - Place where function is maximized (Best Fit) = stack VMs
  - Place where function is minimized (Worst Fit) = spread VMs
- Allows adding custom metrics to be used in placement

Is this what we need in NFV placement?
INTO : Software Based Virtual Integrated Network
NFV + SDN
**Distributed cloud networking = NFV + SDN**

- **Key enablers**
  - Network function virtualization (NFV)
  - Software defined networking (SDN)

- **Ideal for next generation services**
  1) Network services
     - NFV
  2) Automation services
     - Smart X, IoT
  3) Augmented experience
     - Virtual X, Augmented X

---

Placement of Network Functions

- Where to place each function
  - One place (globally)
  - In each location
  - Statically – network planning
  - Dynamically (as needed) depends on demand

- What exactly is
  - The demand
  - The cost (of placing network functions)
  - The constraints (what can be put where)
  - A good placement (objective function)

A network optimization problem
Placement of Network Functions - A Model

Input
- A set of flows, each with a path and a demand for each of the possible network functions.
- A set of datacenters locations, each with a size.
- A set of network functions realizations, each with capacity (amount of clients to be served), size, and establishment cost.

Output
- A placement of copies of the realization of the network functions and a rerouting of the flow into the DCs.

Such that: The demand for each flow and for each function is satisfied, the size constraints are met, and the overall cost is minimal.
Input

- A set of flows, each with a path and a demand for each of the possible network functions.
- A set of datacenters locations, each with a size.
- A set of network functions realizations, each with capacity (amount of clients to be served), size, and establishment cost.

OUTPUT

- A placement of copies of the realization of the network functions and a rerouting of the flow into the DCs.

\[
\begin{align*}
\text{Min} & \quad \sum_{c \in C} \sum_{i \in f(c)} \sum_{u \in U} x_{cu} \cdot d(c, u) + \sum_{u \in U} \sum_{i=1}^{m} y_u \cdot p_u^i \\
\text{s.t.} & \quad \text{for each client } c, \text{ function } i \in f(c): \quad \sum_{u \in U} x_{cu}^i \geq r_c^i, \\
& \quad \text{for each client } c, \text{ node } u, \text{ function } i: \quad x_{cu}^i \leq y_u^i, \\
& \quad \text{for each node } u: \sum_{i=1}^{m} y_u^i \cdot w_u^i \leq w(u), \\
& \quad \text{for each function } i, \text{ node } u: \quad \sum_{c \in C} x_{cu}^i \leq y_u^i \cdot \mu^i, \\
& \quad \text{for each function } i, \text{ node } u: \quad y_u^i = 0 \quad \text{or} \quad y_u^i = 1.
\end{align*}
\]

Such that: The demand for each flow and for each function is satisfied, the size constraints are met, and the overall cost is minimal.
We have to think about the **MODEL**
where the goal is to **optimize REAL SYSTEMS**
Why modeling?

- **Real systems are very complex**
  - Different parameters that affect the result
  - Many configuration options
  - In the Network Function placement case:
    - depends on the actual VNF (vCPE, vCDN, ...)
    - on the underlying infrastructure (VM, container, ...)
    - many more ....

- **Need to capture the important (and only the important) aspects**
  - What is important?
  - How to quantify the affect of these (important) parameters
  - What are the criteria for success (optimization objective)
Addressing an optimization problem

- **Model the problem**
  - must select the “right” perspective
  - this is the most difficult part

- **Find an optimization scheme for the “theoretical” problem**
  - not always so easy
  - most problems are NP-hard
  - approximation or heuristics

- **Apply the solution to the original (real) problem**
  - need to modify the “theoretical” approximation algorithm

- **Evaluate expected performance**
  - in many cases difficult for lack of data (NFV)
Main theoretical result

Theorem:

There exists a bi-criteria ($O(1), O(1)$) approximation algorithm for the General NFV location problem

If we have only a single network function – the problem becomes the **Facility Location Problem**

- classic problem, studied extensively since the 1960s
- NP-hard problem, constant-factor approximation

If the network distances are all zero – the problem becomes the **Generalized Assignment Problem (GAP)**

- Well known problem, studied extensively since the 1990s
- NP-hard problem, constant-factor approximations
Algorithmic Approach

- NFV location problem is computationally hard (NP-hard)
- formulate the problem as an integer program
- relax integrality constraint: obtain a linear program
- compute an optimal fractional solution - lower bound on an optimal integral solution
- round fractional solution into an integral solution, while not increasing objective function by “too much” ...
- bi-criteria approximate solution:
  - size constraints are violated by constant factor
  - objective function value increases by constant factor
NFV Location Problem (uncap.)

for each function i and client c:

- fractional solution induces a probability distribution on the locations from which client c gets service of function i
- \( d(c,i) \) – expected distance; also contribution to LP
- \( B(c,2d(c,i)) \) - ball around c at radius 2\( d(c,i) \)
- it contains at least \( \frac{1}{2} \) the fractions providing service to c. (Markov’s inequality)
- c does not “care” from which location in the ball it gets service
- multiply all fractions by 2:
  - client c gets full service from locations in \( B(c,2(c,i)) \)
  - setup cost is multiplied by 2
  - size constraints are violated by at most a factor of 2
For each function $i$ (separately):

“massage” the solution so that balls (of $i$) are disjoint

- $c^*$ - ball of client having minimum (expected) radius
- $c$ – client receiving non-zero service from the ball of $c^*$
- $c$ is “removed” from the solution and “forgotten”
- eventually, $c$ will get the service of $i$ from the same location as $c^*$, paying at most $2d(c,i) + 4d(c^*,i) \leq 6d(c,i)$.

Recall $d(c,i)$ is the contribution of $c$ to the LP.
NFV Location Problem (uncap.)

- define an instance of GAP:
  - each function $i$ and each “surviving” ball of a client $\rightarrow$ job
  - servers $\rightarrow$ machines
  - fractional solution of NFV location $\rightarrow$ fractional solution to GAP

- round instance of GAP:
  - determines location of functions and mapping of clients
  - setup costs remain the same as in fractional solution to GAP
  - distance costs remain the same as in fractional solution to NFV
  - size of servers is violated by at most a factor of 2

- approximation factor: $(6, 4)$
  - distances multiplied by 6 and setup costs multiplied by 2
  - sizes are multiplied twice by 2
Experimental evaluation

This network covers:

- 195 access locations (mostly within Europe and North America), about 260 links and almost 40 data centers

Input

- A set of flows, each with a path and a demand for each of the possible network functions.
- A set of datacenters locations, each with a size.
- A set of network functions realizations, each with capacity (amount of clients to be served), size, and establishment cost.
- Selected 400 random pairs of (source, destination), and determined a shortest path between each source and destination, unit demand per flow.
- Each such flow is associated with 1-4 network functions that were chosen randomly from a set of 30.
- The size of a network function varies.
- The size of data center was randomly selected in the range 200-500.
- Opening cost was constant.
Experimental evaluation

- Greedy
  - Go over all network function in an arbitrary order
  - For each such function
  - Find in a greedy way the best placement to satisfy the flows’ demand

randomly from a set of 30.
- The size of a network function varies.
- The size of data center was randomly selected in the range 200-500.
- Opening cost was constant.
Experimental evaluation

- **Greedy**
  - Go over all network function in an arbitrary order
  - For each such function
  - Find in a greedy way the best placement to satisfy the flows’ demand randomly from a set of 30.

- The size of a network function varies.
- The size of data center was randomly selected in the range 200-500.
- Opening cost was constant.
How good is this model?

- Service chaining example
  - CPE – FW – DPI

- Can we use the previous model for function placement in this case?

- Can we find a better model?

Source: ETSI Ongoing PoC
How good is this model?

- The order of the functions (per flow) is given
- No pre-defined paths
Service chain model – take 2

• Given
  - Set of services
  - Set of demands

• Find
  - Function placement
  - Flow routing
  - Cloud resource allocation
  - Network resource allocation

• Such that
  - Demands are satisfied
  - Overall operational cost is minimized
cloud network flow
cloud network flow
cloud network flow
cloud network flow
cloud network flow
cloud network flow
cloud network flow
cloud network flow
Service Model

Functions:  
• \((\phi, 1)\)  
• \((\phi, 2)\)  
• \((\phi, M_\phi)\)

Commodities:  
• \((d, \phi, 0)\)  
• \((d, \phi, 1)\)  
• \((d, \phi, M_\phi - 1)\)  
• \((d, \phi, M_\phi)\)

- A network service \(\phi \in \Phi\) is described by a chain of \(M_\phi\) virtual network functions (VNFs)
- \((\phi, i)\) denotes the i-th function of service \(\phi\)
- \((d, \phi, i)\) denotes the output of the i-th function of service \(\phi\) for destination \(d\)
- Function \((\phi, i)\) has resource requirement \(r^{(\phi, i)}\) processing resource units per flow unit, scaling factor \(\xi^{(\phi, i)}\) output flow units per input flow unit
Service chain model – take 2

\[
\min \sum_{(u,v)} w_{uv} y_{uv}
\]

s.t.

\[
\sum_{v \in \delta^- (u)} f^{(d, \phi, i)}_{uv} = \sum_{v \in \delta^+ (u)} f^{(d, \phi, i)}_{uv} \quad \forall u, d, \phi, i
\]

\[
f^{(d, \phi, i)}_{p(u), u} = \xi (\phi, i) f^{(d, \phi, i-1)}_{u, p(u)} \quad \forall u, d, \phi, i
\]

\[
\sum_{(d, \phi, i)} f^{(d, \phi, i)}_{uv} \leq y_{uv} \leq c_{uv} \quad \forall (u, v)
\]

\[
f^{(d, \phi, 0)}_{s(u), u} = \lambda_u (d, \phi) \quad \forall u, d, \phi
\]

\[
f^{(d, \phi, M_{\phi})}_{u, a(u)} = 0 \quad \forall d, \phi, u \neq d
\]

\[
f^{(d, \phi, i)}_{uv} \geq 0, \ y_{uv} \in \mathbb{Z}^+ \quad \forall (u, v), d, \phi, i
\]

Cost Function

Combined Flow Conservation

Service Chaining

Capacity

Sources and Demands

Fractional flows

Integer resources
Service chain model – take 2

\[ \min \sum_{(u,v)} w_{uv} y_{uv} \]

s.t.

\[ \sum_{v \in \delta^{-}(u)} f^{(d, \phi, i)}_{v, u} = \sum_{v \in \delta^{+}(u)} f^{(d, \phi, i)}_{u, v} \quad \forall u, d, \phi, i \]

\[ f^{(d, \phi, i)}_{p(u), u} = \xi^{(\phi, i)} f^{(d, \phi, i-1)}_{u, p(u)} \quad \forall u, d, \phi, i \]

\[ \sum_{(d, \phi, i)} f^{(d, \phi, i)}_{u, v} \leq y_{uv} \leq c_{uv} \quad \forall (u, v) \]

\[ f^{(d, \phi, 0)}_{s(u), u} = \lambda^{(d, \phi)}_u \quad \forall u, d, \phi \]

\[ f^{(d, \phi, M_{\phi})}_{u, u} = 0 \quad \forall d, \phi, u \neq d \]

\[ f^{(d, \phi, i)}_{u, v} \geq 0, \ y_{uv} \in \mathbb{Z}^+ \quad \forall (u, v), d, \phi, i \]
Main Theoretical Result

There is a fast approximation algorithm for the fractional NSDP that produces an $\epsilon$ approximation solution in time $O(m^2nL/\epsilon)$.

Use dynamic evolution of underlying queuing system to construct an iterative approximation to original static problem.

Performance

![Diagram with arrows and nodes]

- $r = 1$  
- $r = 3$
- $r = 2$
- $r = 2$

![Graphs and charts]

- Cost vs. Iterations
- Flow Conservation Violation
- Processing Flow Rate

- DCNC, $V=40$
- QNSD, $V=40$, $\theta = 0$
- QNSD, $V=150$, $\theta = 0.9$

- Service 1, Function 1
- Service 1, Function 2
- Service 2, Function 1
- Service 2, Function 2

Node Index:

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
How good is this model?

• Previous models address placement in node (DC) granularity

• How about physical host granularity?

• Placement of VNF VMs in the physical hosts

Source: ETSI Ongoing PoC
So what did we do today?

- Resource allocation
- Placement in the Cloud
  - multidimensional dual bin packing
- Placement in Open Stack
- Placement of NF chaining
  - first model
  - second model
- Is it good enough
  - to be used in practice
- Next week – yet a different model and more on implementation details and performance