NetSlicer: Automated and Traffic-Pattern Based Application Clustering in Datacenters

Liron Schiff¹, Ofri Ziv¹, Manfred Jaeger² and Stefan Schmid^{2,3}













Agenda

The problem

- Our ML approach
- Challenges we encountered
- The NetSlicer algorithm
 - Evaluation
- Use case



۰.



The network is the sum of its applications





Leveraging existing data to group nodes

Many definitions, many sources

Orchestration (e.g., tags, names, desc.)

Inventory (e.g., CMDB)

DevOps (e.g., puppet recipes, ansible scripts)

Navigator	🖡 📅 hr-db-01 🛛 Actions 👻			
Home 🕥	Getting Started Summary	Monitor Manage Related Objects		
✓	 VM Hardware 			
➡ Datacenter Site A	► CPU 1 CPU	I(s), 0 MHz used	_	
♥ III Cluster Site A	Memory 512	2 MB, 5 MB memory active		
esx-01a.corp.iocal	Hard disk 1 20.00	GB		
ESX Agents	Network adapter 1 Manag	gement Network (connected)		
hr-db-01	> Network adapter 2 yow-d	/s-35-vidualwire-3-sid-5001-LS-HR-01 (conn	nected)	
hr-web-01				
hr-web-02	CD/DVD drive 1 Discon	nnected	. 0	
legal-db-01	Floppy drive 1 Discor	nnected 🧨	- 0	
legal-web-01	▶ Video card 4.00 M	IB		
mkto-db-01	Other Addition	onal Hardware		
mktg-web-01	Compatibility ESXi 5	5.5 and later (VM version 10)		
🗕 🗊 Management & Edge Clu	i			
es 🎁 Services	 Resource Groups Resource Groups 	FIAM DEC2 DE EC2 Container Service	RDS	∤ Elastic Bea
es Es Esta			100	
ES EC2 Dashboard	Launch Instance Connect A	ctions *		
Tags NS	Q. Filter by tags and attributes or sear	ch by keyword		
NS Reports	Q. Filter by tags and attributes or sear	ch by keyword		Instance Type
NS Limits	Filter by tags and attributes or sear Name server1.test.com	ch by keyword V Instance ID i-01b7a10/8ed1a/7e8	3	Instance Type
Tags Reports Limits	C. Filter by tags and attributes or sear Name server1.test.com server2.test.com	ch by keyword Instance ID i-01b7a10f8ed1af7e8 i-09f55cb9133ce5710	3 0	t2.small t2.micro
NS Tags Reports Limits INSTANCES Instances Spot Requests	Filter by tags and attributes or sear Name server1.test.com server2.test.com server3.test.com	ch by keyword	3 0 9	Instance Type 12.small t2.micro t2.micro
Tags Reports Limits Instances Spot Requests Reserved Instances Scheduled Instance	Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server3.test.com	ch by keyword	- 3 9	Instance Type t2.small t2.micro t2.micro
Tags Reports Limits Inits Instances Spot Requests Reserved Instance Scheduled Instance Dedicated Hosts	Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server3.test.com Instance: j I-01b7a10f8ed1af7e8 (sec	ch by keyword	3 0 19	r Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Imits Instances Spot Requests Reserved Instance Scheduled Instance Dedicated Hosts	Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server3.test.com Instance: j1-01b7a10f8ed1af7e8 (see Description Status Checks	ch by keyword	3 0 9	Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Inits INSTANCES Instances Spot Requests Reserved Instance Scheduled Instance Dedicated Hosts Interest Dedicated Hosts Interest Dedicated Hosts	Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server3.test.com Instance: I-01b7a10f8ed1af7e8 (see Description Status Checks	ch by keyword	3 0 9	Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Inits INSTANCES Instances Spot Requests Reserved Instances Scheduled Instance Scheduled Instance Dedicated Hosts IMAGES AMIs Bundle Tasks I Batto Block et al.	Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server2.test.com Instance: I-01b7a10f8ed1af7e8 (sc Description Status Checks Add/Edit Tags	ch by keyword	3 0 9	Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Imits Instances Spot Requests Reserved Instances Scheduled Instance Dedicated Hosts ImMAGES AMIs Bundle Tasks ImAGES AMIs Bundle Tasks	RE Filter by tags and attributes or sear Name Server1.test.com Server2.test.com Server2.test.com Instance: I-01b7a10f8ed1af7e8 (sec Description Status Checks Add/Edit Tage Key	ch by keyword	3 0 9 Value	Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Imits Instances Spot Requests Reserved Instances Scheduled Instance Dedicated Hosts Images AMIs Bundio Tasks ELASTIC BLOCK STO Volumes Snapshots	RE RE Key Name	ch by keyword	3 0 9 Value server1.test.com	Instance Type 12.small 12.micro 12.micro
Tags Reports Limits Imits Imits Instances Spot Requests Reserved Instances Scheduled Instance Scheduled Instance Dedicated Hosts Images AMIs Bundie Tasks ELASTIC BLOCK STO Volumes Snapshots	RE Add/Edit Tage Key Name	ch by keyword	3 0 9 Value server1 AssLcom server1	Instance Type 2 12.small 12.micro



Leveraging existing data to group nodes

Many definitions, many sources

Orchestration (e.g., tags, names, desc.)

Inventory (e.g., CMDB)

DevOps (e.g., puppet recipes, ansible scripts)

Limited by

Availability (not always defined)

Relevance (may not be up to date)

Consistency (may not be standardized)









GuardiCore

Big-DAMA 2018

Cluster nodes into tiers





8

- Cluster nodes into tiers
- Cluster tiers into apps





Cluster nodes into tiers

- Cluster tiers into apps
 - Identify roles (Infras, services)



- Cluster nodes into tiers
- Cluster tiers into apps Identify roles (Infras, services)
- Generate (simple) labels
 - Based on services
 - Based on machine names







Example - ungrouped



12

Example - grouped by application





GuardiCore





Big-DAMA 2018

Load balancing

- Same tier nodes are not "identical"
- Different servers may serve different clients





Load balancing

GuardiCore

Infras and monitors

- Add "noisy" connections
- Over-connected network





Load balancing

GuardiCore

Infras and monitors

Non-standard ports

- Customized and random port numbers
- Harder to identify apps and filter noise



Services

Certificate Services (CertSvc) Distributed File System (Dfs) Event Log Exchange Server Fax Service File Replication Local Security Authority (LSASS) Netlogon Remote Storage Terminal Services ...

Protocols
DCOM FTP HADOOP RPC
•••

- Load balancing
- Infras and monitors
- Non-standard ports
- Uncovered nodes

GuardiCore

Missing connection / L7 data





- Load balancing
- Infras and monitors
- Non-standard ports
- Uncovered nodes
 - Missing connection / L7 data





- Load balancing
- Infras and monitors
- Non-standard ports
- Uncovered nodes
- Big data
 - Tens of thousands of nodes (and millions of links)
 - (almost) unlimited external internet hosts

 $\textbf{Table 2.} \ \textbf{Number of data centres by size category and growth rates}$

Data centres category	No of data centres (2013)	Change in No of data centres in 2008- 2013
Server Cabinets (3-10 m ²)	30500	-8%
Server Rooms (11-100 m ²)	18100	+/- 0%
Small Data centres (101-500 m ²)	2150	+23%
Medium Data centres (501-5000 m ²)	280	+27%
Large Data centres (over 5000 m ²)	70	+40%
Sources Hintomann and Clauson 2014		





Figure 2: Development of the numbers of physical servers and virtual servers in data centers in Germany (Source: calculations by Borderstep on the basis of Techconsult eanalyzer)







Start with an annotated graph

each link represents connections at a single port



Merge nodes at 3 phases:



Merge nodes at 3 phases:

Detect "clusters"

Strongly connected components



GuardiCore

BIG-DAMA 2018

Merge nodes at 3 phases:

Detect "clusters"

Strongly connected components

Detect tiers

based on neighborhood similarity

Detect applications

based on weighted connectivity



GuardiCore

BIG-DAMA 2018

Detecting Tiers







Endpoint definition $(u,p,t) \in \Phi$:

u: a node id

- *p*: a service port number
- *t*: type (ⓒ:client / ⑤:server)



n	ode id	connected endpoints
	Х	$(a, 80, \bigcirc), (b, 80, \bigcirc), (d, 21, \bigcirc), (e, 21, \bigcirc)$
	У	$(c, 23, \mathbb{C}), (b, 80, \mathbb{C}), (f, 21, \mathbb{S}), (e, 21, \mathbb{S})$
	b	(x,80,\$), (y,80,\$)

Endpoint definition $(u,p,t) \in \Phi$:

- *u*: a node id
- p: a service port number
- t: type (©:client / ③:server)

Weight function $W_v: \Phi \rightarrow R+$

- set weights to endpoint with respect to v
- E.g., $W_v(\phi)=1$ if ϕ connected with v, o.w. 0



node id	connected endpoints
х	(a,80,ⓒ), (b,80,ⓒ), (d,21,⑤), (e,21,⑤)
У	$(c,23, \bigcirc), (b,80, \bigcirc), (f,21, \bigcirc), (e,21, \bigcirc)$
d	(x,80,S), (y,80,S)



Similarity by normalized dot product:

$$Sim(u,v) = \sum_{\phi \in \Phi} W_u(\phi) W_v(\phi) / (|W_u| \cdot |W_v|),$$

where
$$|W_x| = \sqrt{\sum_{\phi \in \Phi} W_x(\phi)^2}$$
.



node id	connected e	endpoints		
х	(a,80,©),	(b,80,©)	, (d,21,\$)	(e,21,§)
У	(c,23,©),	(b,80,©)	, (f,21,\$)	(e,21,§)
d	(x,80,\$),	(y,80,S)		

Similarity by normalized dot product:

• $Sim(u, v) = \sum_{\phi \in \Phi} W_u(\phi) W_v(\phi) / (|W_u| \cdot |W_v|),$

where
$$|W_x| = \sqrt{\sum_{\phi \in \Phi} W_x(\phi)^2}$$
.

Consider noisiness of endpoints

- noise(ϕ) = max (1 Sim(u,v)) connected to ϕ
- $W_v(\phi) = (1 \text{noise}(\phi))^{\alpha}$



Similarity by normalized dot product:

•
$$Sim(u, v) = \sum_{\phi \in \Phi} W_u(\phi) W_v(\phi) / (|W_u| \cdot |W_v|),$$

where
$$|W_x| = \sqrt{\sum_{\phi \in \Phi} W_x(\phi)^2}$$
.

Consider noisiness of endpoints

noise(ϕ) = max (1 - Sim(u,v)) connected to ϕ

•
$$W_{v}(\phi) = (1 - \text{noise}(\phi))^{\alpha}$$

÷.

GuardiCore

Similarity and noise are computed in several rounds



Detecting Tiers

An iterative process

- computes similarity and noise
- merges similar nodes
- stops when no similar pair found









From tiers to applications

Start from graph of tier nodes





34

- Start from graph of tier nodes
- Assign weights to (merged) links





- Start from graph of tier nodes
- Assign weights to (merged) links
- Ignore weak/infra links





- Start from graph of tier nodes
- Assign weights to (merged) links
- Ignore weak/infra links
- Group connected components





Implementation

- Programed in Python
- Parallel design
- Optional initial noise configuration
 - for better and faster results
 - e.g., super noisy ports and nodes
- Modular
 - any clustering alg. for tiers and apps (given similarity values)



÷.

Evaluation

Datasets:

- 3 datacenter networks
- monitored at core VMs
- ground truth for monitored
- <u>available online</u>

	net1	net2	net3
dominat OS	۵	4	4
#nodes	3003	2507	4588
#monitored	153	143	174
#unmonitored	2850	2364	4414
ground-truth apps	40	58	85



Distribution of nodes to app sizes

Evaluation

Datasets:

- 3 datacenter networks
- monitored at core VMs
- ground truth for monitored
- available online

Compared with:

- louvain modularity
- node2vec (+ HAC/HDBSCAN)
- and mixtures

Scoring:

Adjusted Random Index (ARI) from ground truth

	net1	net2	net3
dominat OS	۵	4	4
#nodes	3003	2507	4588
#monitored	153	143	174
#unmonitored	2850	2364	4414
ground-truth apps	40	58	85



Distribution of nodes to app sizes

Ċ.

BIG-DAMA 2018

Results





Analyst Compatible







BIG-DAMA 2018

44



Big-DAMA 2018



🔇 GuardiCore

BIG-DAMA 2018

The NetSlicer Algorithm

Customized similarity

Considers real life scenarios

Promising initial results

Parallel and modular design





÷.

47

Future work

More datasets

Clustering processes

(multiple apps per server)

Mixing with more clustering algs.

Convergence time bounds





Questions?



datasets: <u>https://www.guardicore.com/labs/datacenter-traces/</u> more works: <u>https://www.guardicore.com/labs/research-academic/</u> me: liron.schiff@guardicore.com

Architecture



)re 50