SISTEDES 2019
Jornadas de la Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software
Del 2 al 4 de septiembre de 2019, Cáceres, Extremadura, España

Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications

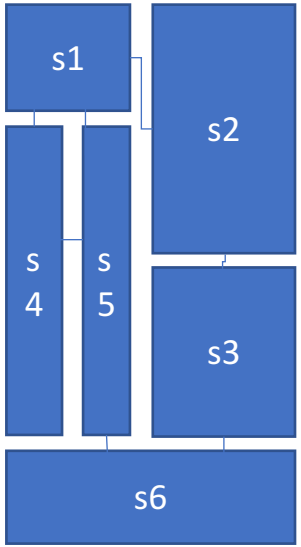
Publicado en Journal of Supercomputing (2018), Q2,
https://doi.org/10.1007/s11227-018-2345-2

Carlos Guerrero, **Isaac Lera**, Carlos Juiz
Department of Computer Science,
University of Balearic Islands, Spain
isaac.lera@uib.es

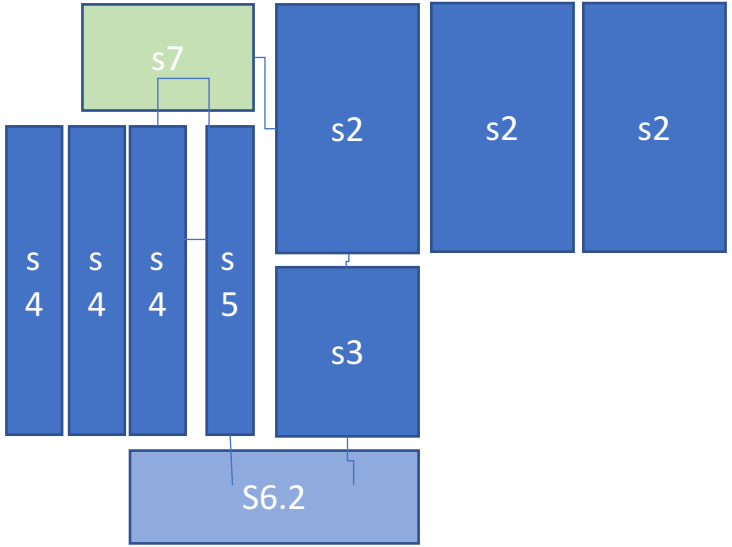
Universitat de les
Illes Balears

¿En qué consiste una aplicación basada en microservicios?

Es un estilo de desarrollo



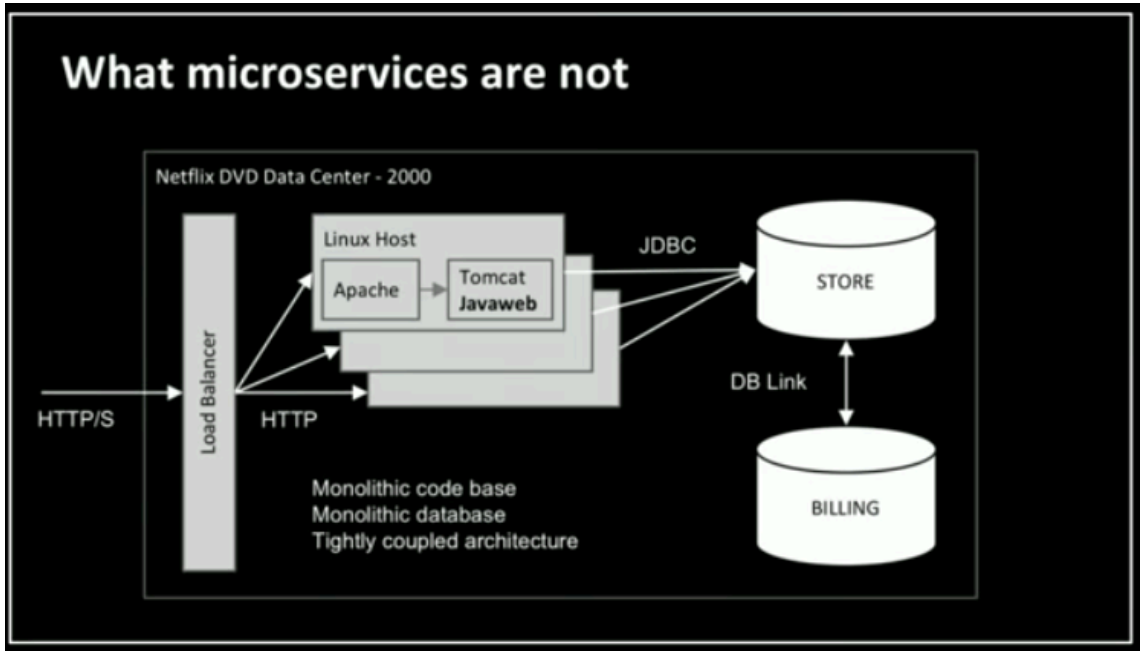
Pequeños
Independientes
Comunicación por mensajes

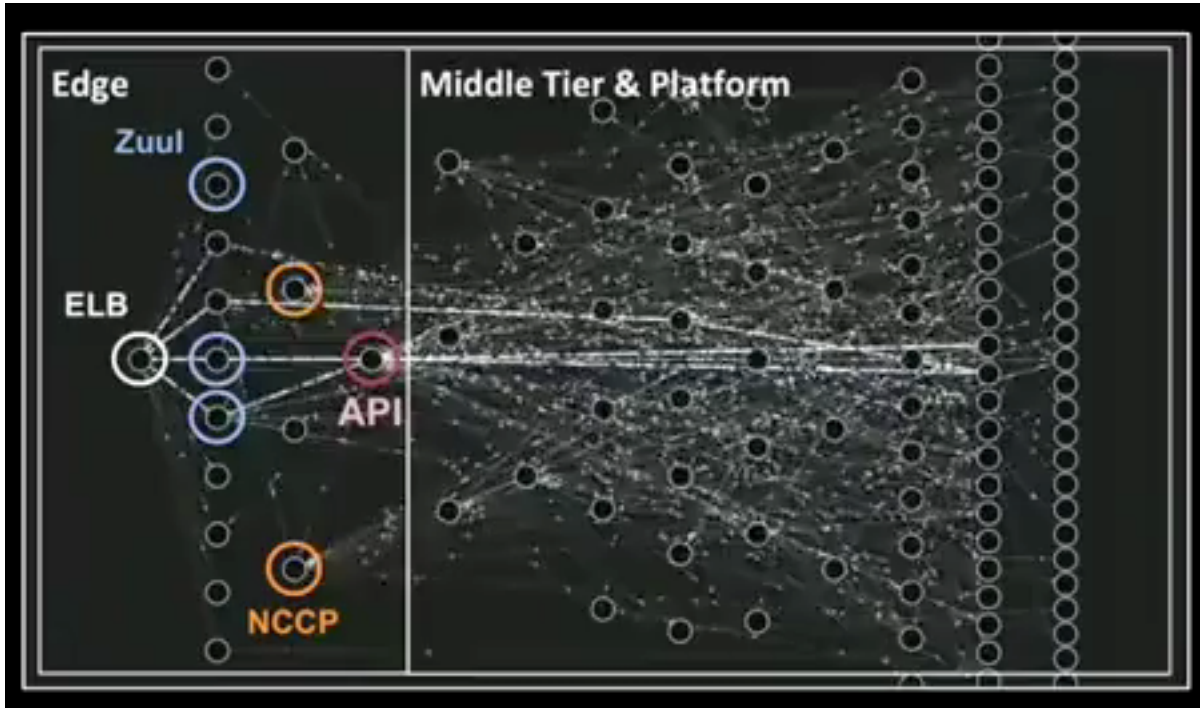


Beneficios:

- Separación de conceptos: “Ciclo de vida” independiente
- Escalado horizontal o por particionamiento de carga de trabajo
- Virtualización y elasticidad

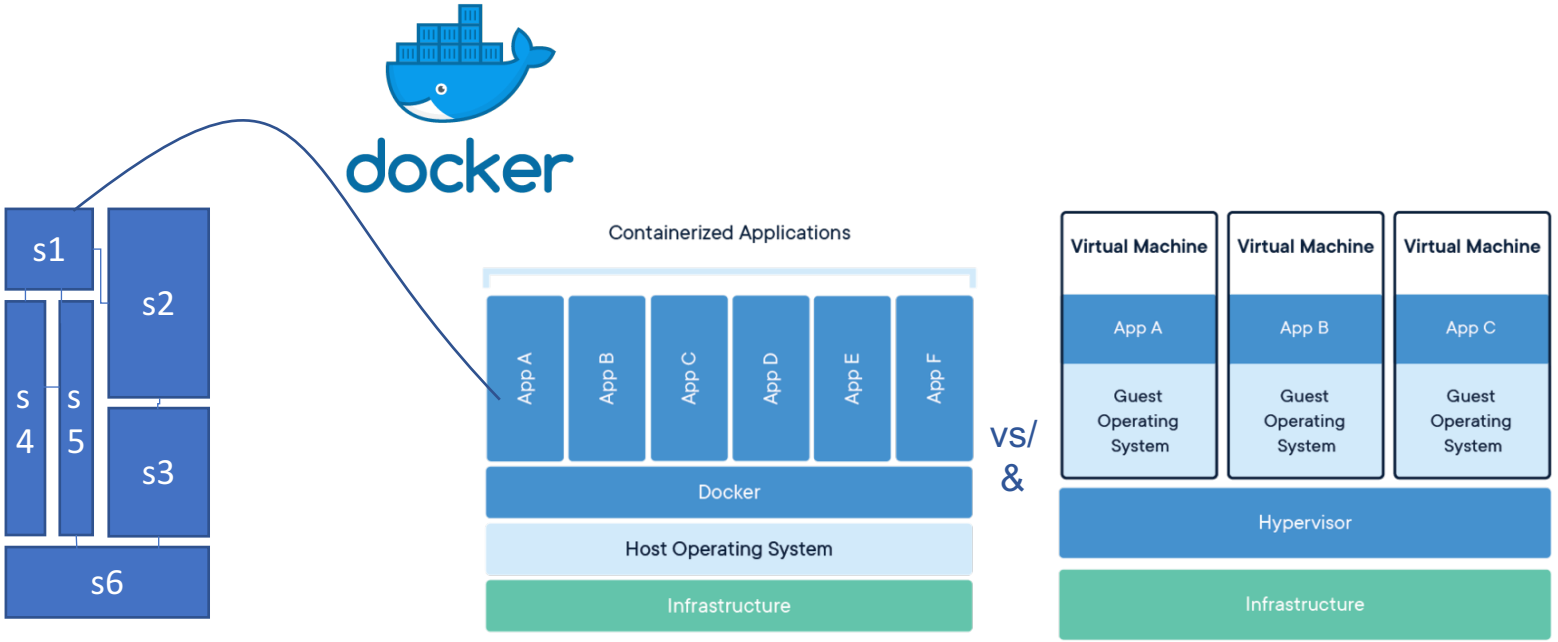
Martin Fowler
https://martinfowler.com/articles/microservices.html


What microservices are not
Netflix DVD Data Center - 2000
Linux Host
Apache
Tomcat
Javaweb
JDBC
STORE
DB Link
BILLING
Load Balancer
HTTP/S
HTTP
Monolithic code base
Monolithic database
Tightly coupled architecture



Mastering Chaos - A Netflix Guide to Microservices

Cada microservicio se suele desplegar en un "Container"



https://www.docker.com

DESPLIEGUE

Amazon EKS

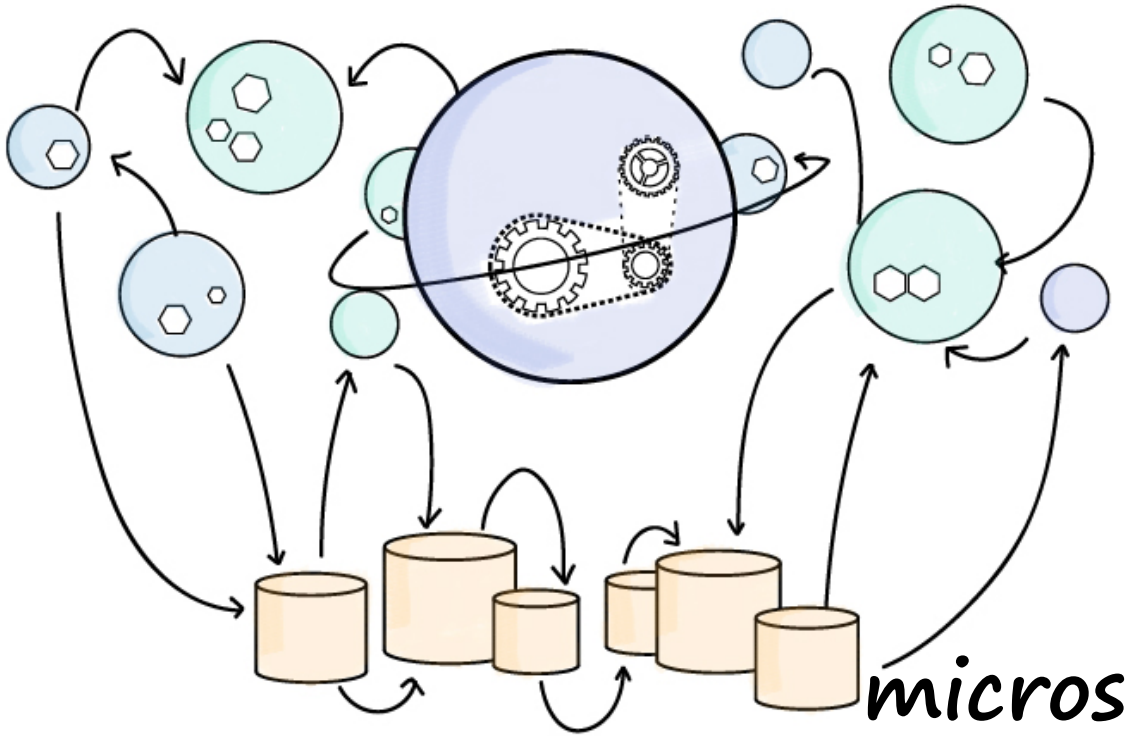
Amazon EKS makes it easy to run Kubernetes on AWS

Google Cloud

Sistemas de Software altamente distribuidos

multi-componente

infraestructura



microservicios

How to Place Your Apps in the Fog - State of the Art and Open Challenges
Antonio Brogi, Stefano Forti, Carlos Guerrero, Isaac Lera
https://arxiv.org/abs/1901.05717

Cumplimiento de Requerimientos no funcionales

NP-hard

Nuestra pregunta de investigación

¿

Podemos implementar una multi-optimización de recursos en la orquestación de containers en multi-cloud, que minimice:

1. El coste económico del despliegue 
2. La latencia de red entre las comunicaciones de microservicios 
3. El tiempo de recuperación en caso de fallo de una instancia 

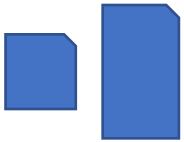
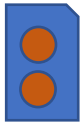
?

Comparar dos alternativas de optimización:

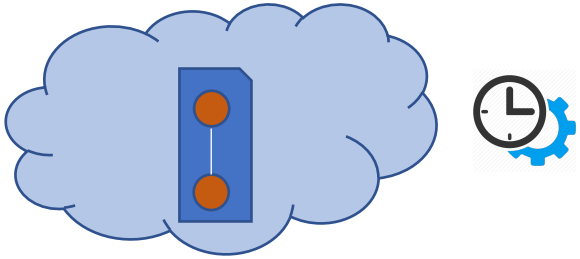
a) un algoritmo voraz (greedy) y

b) un algoritmo evolutivo: Non-dominated Sorting Genetic Algorithm II (NSGA-II)

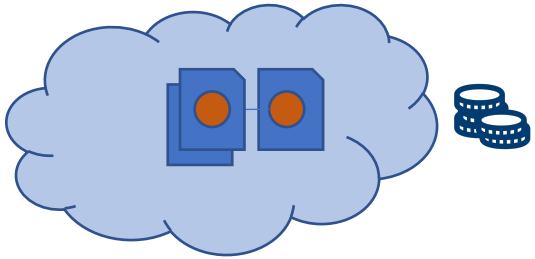
Supuesto

- Provedores de cloud 
- Cada proveedor ofrece diferentes plantillas de VM, *<coste,recursos>* 
- Los microservicios se encapsulan en containers que a su vez se despliegan en VM. 

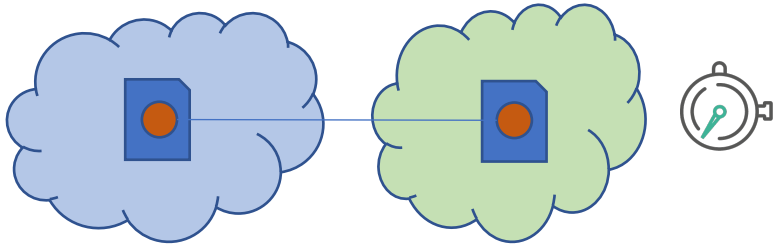
Situación A



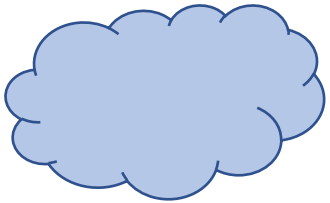
Situación B



Situación C



Definición del problema



- Proveedor de cloud: $prov_n$
- Latencia entre prov.: $L_{n,n'}$
- Tipos de VM: $VMT_{n,i}$
- Coste de VM: Cn,i

- Instancia de VM:$vm^j_{n,i}$
- Recursos disponibles: $R^{avail}_{n,i}$
- Utilización: $U[vmi^j_{n,i}]$

- Microservicio *y* para app *x*: $ms_{x,y}$
- Instancia MS: $msi^z_{x,y}$
- Recursos computacional y almacenamiento: $R^{req}_{x,y.}$ Y $R^{str}_{x,y}$
- $T^{start}_{x,y}$ – tiempo de arranque
- $T^{dwnld}_{x,y}$ – tiempo de descarga
- T – reparación en VM y en Proveedor

$alloc[msi^z_{x,y}]$ $store[msi^z_{x,y}]$ – Ubicación del MS y Ubicación del VM

Calculo del coste

$$C_{total} = \sum_{vmi_{n,i}^{j}}^{j} C_{n,i} \times U\left[vmi_{n,i}^{j}\right]$$

Calculo de la latencia

$$L_{total} = \sum_{msi_{x,y}^{z}}^{z} \left[\sum_{ms_{x,y'} \in MS_{x,y}^{cons}} min(L_{n,n'}) \right]$$

$$as\ alloc\left[msi_{x,y}^{z}\right] = prov_n \wedge prov_{n'} = alloc\left[msi_{x,y'}^{z'}\right] \forall\, msi_{x,y'}^{z'} \in MS_{x,y}^{cons} \tag{3}$$

Calculo del tiempo de reparación

$$T^{repair} = \sum_{msi^{z}_{x,y})}^{z} \left[T^{repair}_{vm} \left[msi^{z}_{x,y} \right] + T^{repair}_{provider} \left[msi^{z}_{x,y} \right] \right] \tag{4}$$

$$T^{repair}_{vm} \left[msi^{z}_{x,y} \right] = \begin{cases} 0.0, & if \quad \exists\, msi^{z}_{x,y} \ \wedge \ msi^{z'}_{x,y} \\ & \mid alloc \left[msi^{z}_{x,y} \right] \neq alloc \left[msi^{z'}_{x,y} \right] \\ T^{start}_{x,y}, & if \quad \exists\, msi^{z}_{x,y} \ \wedge \ msi^{z'}_{x,y} \\ & \mid alloc \left[msi^{z}_{x,y} \right] \neq store \left[msi^{z'}_{x,y} \right] \\ T^{start}_{x,y} + T^{dwnld}_{x,y}, & otherwise \end{cases} \tag{5}$$

Optimización múltiple

By minimizing:

$$C_{total} \wedge L_{total} \wedge T^{repair}$$

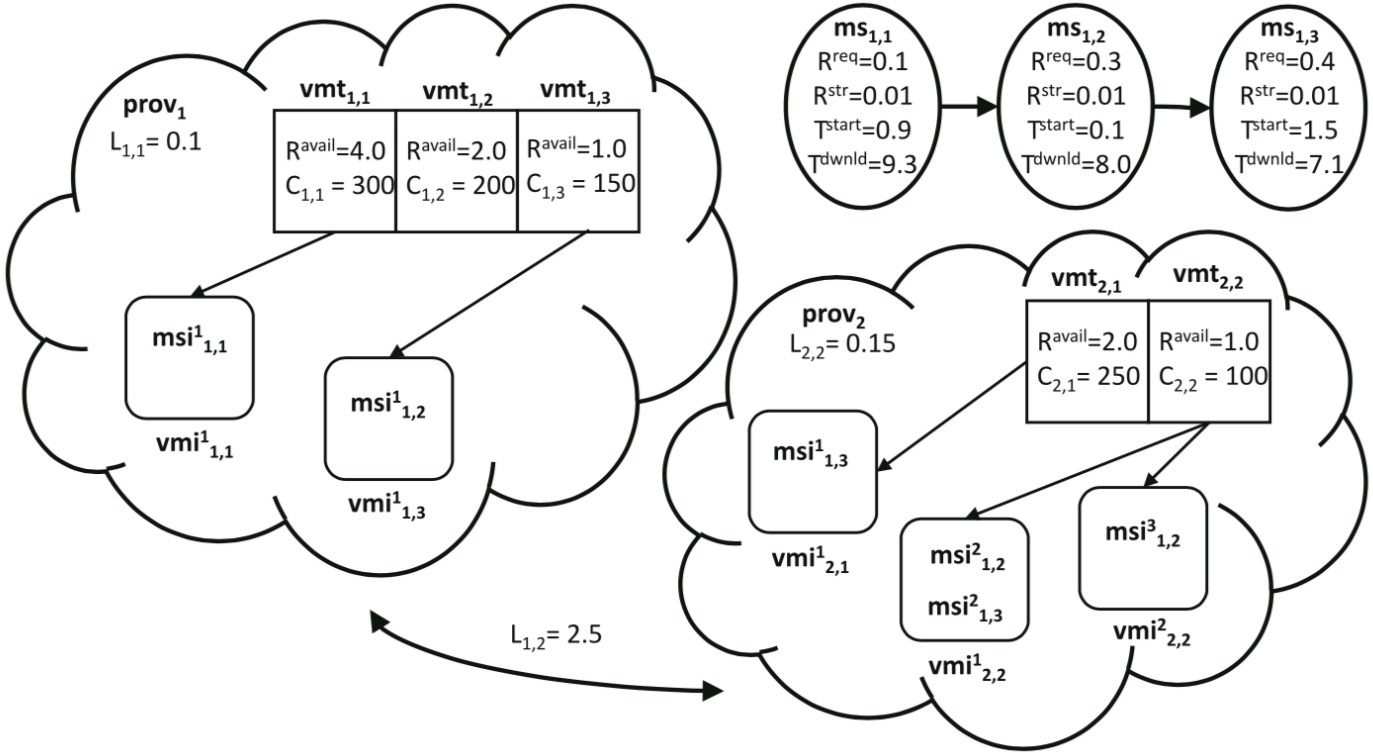
Subject to the constraints:

$$size\left(|msi^{z}_{x,y}|\right) > 1 \quad \forall\, ms_{x,y}$$

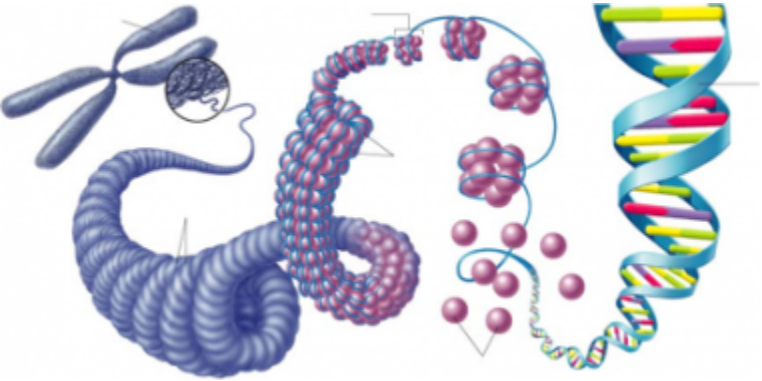
$$U[vmi_{j}(n,i)] < R^{avail}_{n,i} \quad \forall\, vmi^{j}_{n,i}$$

$$U\left[vmi^{j}_{n,i}\right] = \sum_{alloc[msi^{z}_{x,y}]=vmi^{j}_{n,i}} R^{req}_{x,y} + \sum_{store[msi^{z}_{x,y}]=vmi^{j}_{n,i}} R^{str}_{x,y}$$

Supuesto reformulado



Multioptimización (A)



Algoritmos Genéticos

Propuestas (meta-)heurísticas para obtener rankings de despliegues elegibles como soluciones al problema de optimización

Algorithm 1 Multi-objective optimization algorithm [12]

```
1: procedure NSGA- II
2:     populationSize ← 200
3:     generationNumber ← 250
4:     mutationProb ← 0.25
5:     P_t ← generateRandomPopulation(populationSize)
6:     fitness ← calculateFitness(P_t)
7:     fronts ← calculateFronts(P_t, fitness)
8:     distances ← calculateCrowding(P_t, fronts, fitness)
9:     for i < generationNumber do
10:        P_off = ∅
11:        for j < populationSize do
12:            father1, father2 ← binaryTournamentSelect(P_t, fronts, distances)
13:            child1, child2 ← crossover(father1, father2)
14:            if random() < mutationProb then mutate(child1),mutate(child2)
15:            P_off = P_off ∪ {child1, child2}
16:        P_off = P_off ∪ P_t
17:        fitness ← calculateFitness(P_t)
18:        fronts ← calculateFronts(P_off, fitness)
19:        distances ← calculateCrowding(P_off, fronts, fitness)
20:        P_off = orderElements(P_off, fronts, distances)
21:        P_t = P_off[1..populationSize] #the best half
22:    Solution = fronts[1] #the Pareto front
```

Población inicial

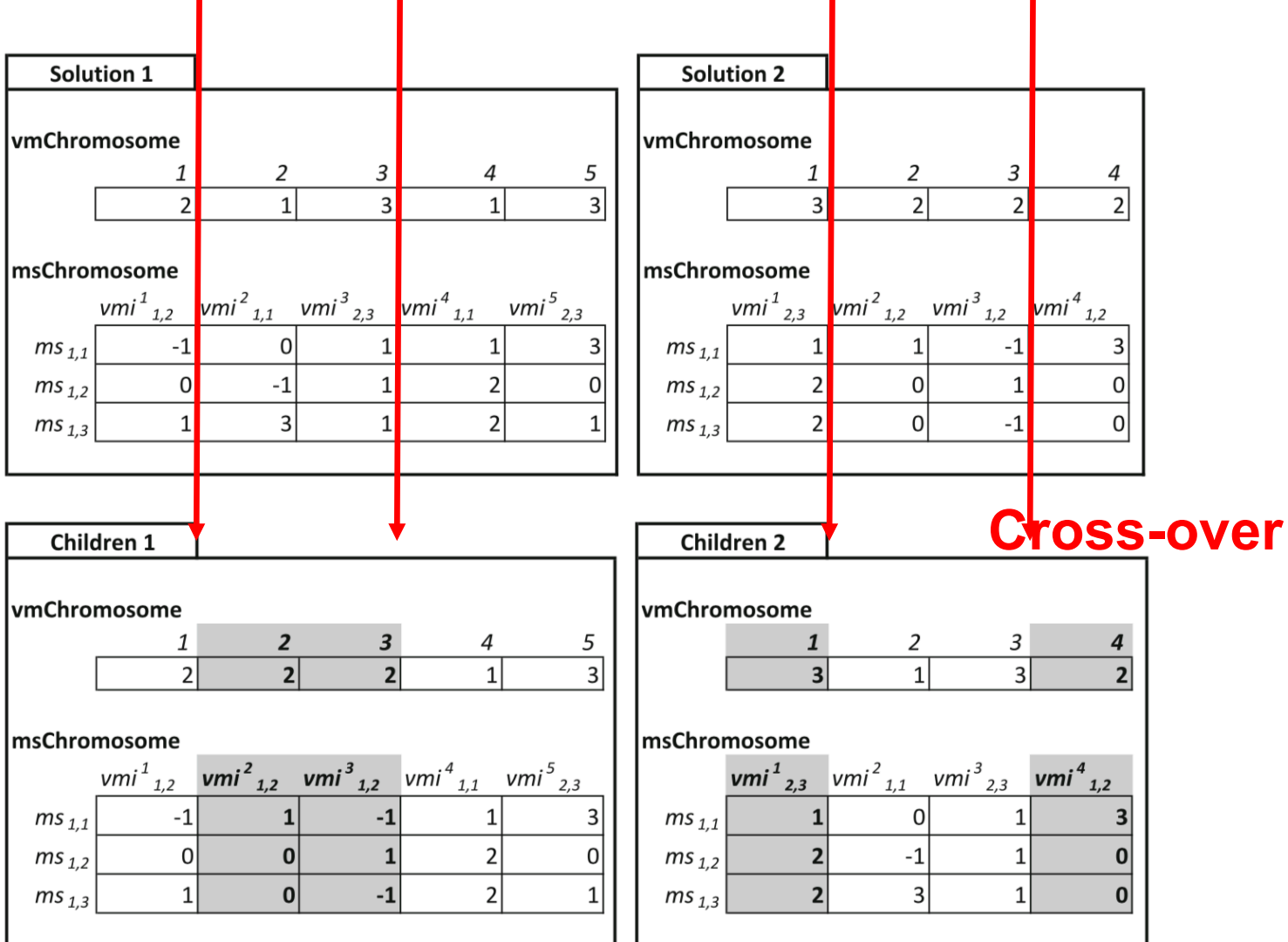
Selección de emparejamientos

Cruce

Mutaciones

Supervivencia de hijos

Supervivencia de individuos



Función Fitness

Dado un conjunto de asignaciones de microservicios y VM sobre una infrastructura:

$$\sum_{i}^{num.\,objectives} \omega_i \times \theta_i \times i$$

$$= \frac{1}{3} \times \theta_{C_{total}} \times C_{total} + \frac{1}{3} \times \theta_{L_{total}} \times L_{total} + \frac{1}{3} \times \theta_{T^{repair}} \times T^{repair}$$

Multioptimización (B)

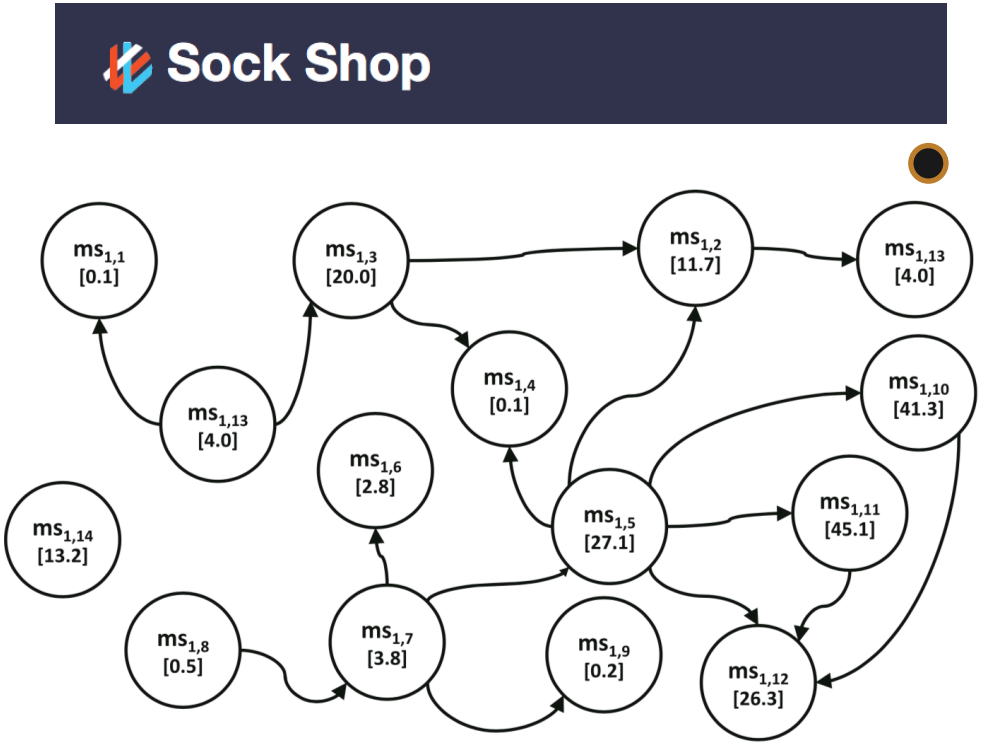
Algoritmos Voraces

Propuestas (meta-)heurísticas donde en cada paso local se selecciona la solución óptima con la esperanza de obtener una solución óptima a nivel general

Algorithm 2 Greedy First-Fit algorithm

```
1: procedure FINDALLOCATION(ms)
2:     for vm in |vmi^j_{n,i}| do
3:         if usage(vm) + resources(ms) < capacity(vm) then return vm
4:     if not allocated then
5:         vmTypePreference = weightedSumObjectiveEvaluation(ms, |vmt_{n,i}|)
6:         bestVmType = min(vmTypePreference)
7:         return newVMInstance(bestVmType)
8: procedure GREEDY
9:     solution = ∅
10:    for ms in shuffle(|ms_{x,y}|) do
11:        vm = findAllocation(ms)
12:        solution+ = allocation(ms, vm)
13:    scalingLimit = 1000
14:    incrementalSolution = solution
15:    for 1 to scalingLimit do
16:        ms = random(|ms_{x,y}|)
17:        vm = findAllocation(ms)
18:        incrementalSolution+ = random[allocation(ms, vm) or store(ms, vm)]
19:        if fitness(solution) > fitness(incrementalSolution) then
20:            solution = incrementalSolution
21:    return solution
```

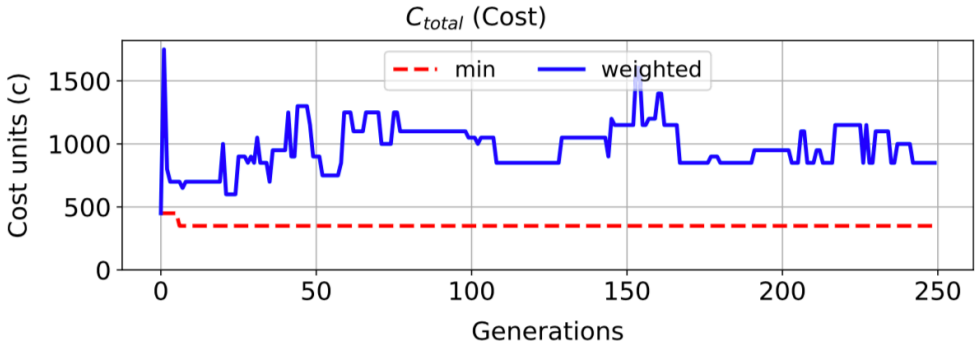
Evaluación experimental

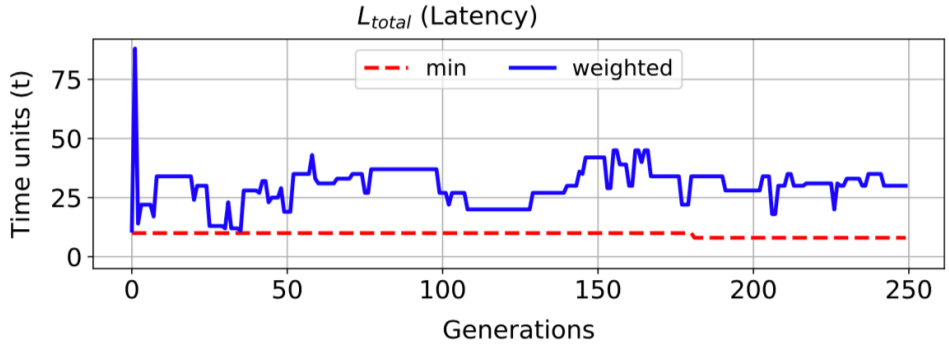


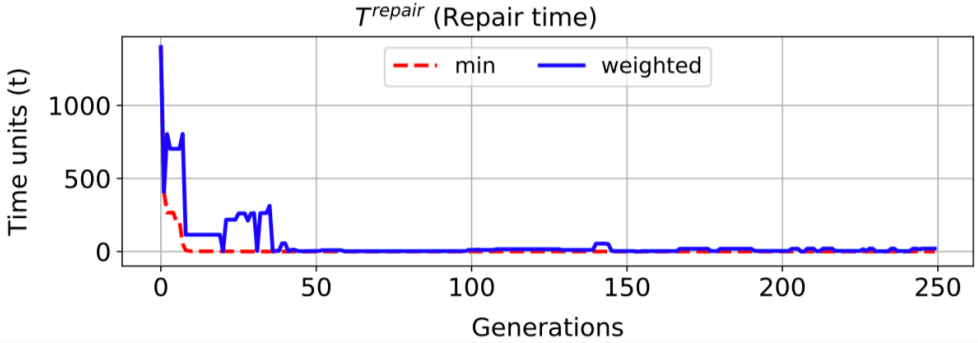
VM type	Provider $prov_n$	Capacity (%cpu) $R_{n,i}^{avail}$	Cost (c) $C_{n,i}$
$vmt_{1,1}$	1	100.0	100.0
$vmt_{1,2}$	1	200.0	150.0
$vmt_{2,3}$	2	400.0	250.0
$vmt_{3,4}$	3	800.0	1000.0

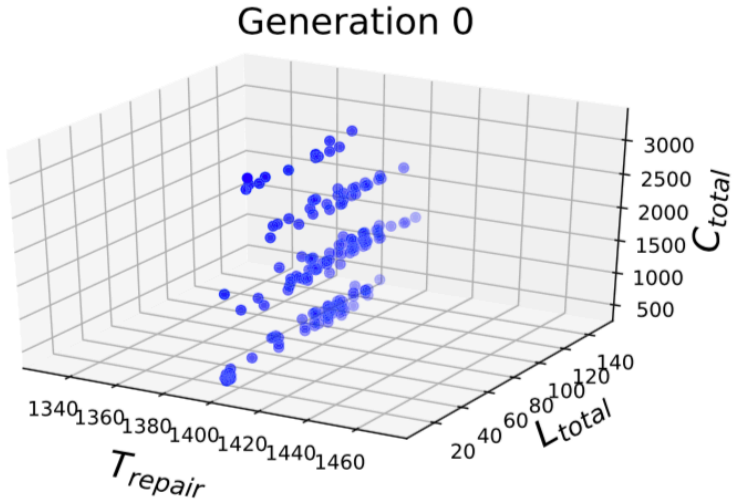
https://microservices-demo.github.io/

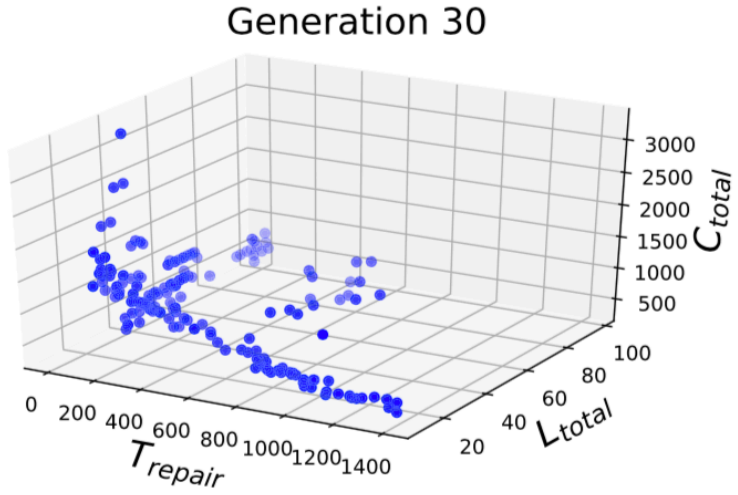
Parameter multipliers			Greedy			NSGA-II		
$C_{n,i}$	$R_{n,i}^{avail}$	$L_{n,n'}$	T^{repair} (t)	L_{total} (t)	C_{total} (c)	T^{repair} (t)	L_{total} (t)	C_{total} (c)
x1.0	x1.0	x1.0	1400	0	250	22	33	800
x2.0	x1.0	x1.0	0	65	4700	20	24	1600
x3.0	x1.0	x1.0	0	74	13,500	2	35	2850
x4.0	x1.0	x1.0	0	43	15,400	19	6	7600
x5.0	x1.0	x1.0	0	48	14,250	22	8	4750
x1.0	x2.0	x1.0	0	41	1300	0	23	950
x1.0	x3.0	x1.0	0	28	850	52	0	500
x1.0	x4.0	x1.0	700	0	500	12	0	700
x1.0	x5.0	x1.0	700	0	500	21	29	700
x1.0	x1.0	x2.0	0	106	1750	0	16	950
x1.0	x1.0	x3.0	0	114	2850	16	66	1850
x1.0	x1.0	x4.0	1400	0	250	24	20	750
x1.0	x1.0	x5.0	1400	0	250	4	45	850

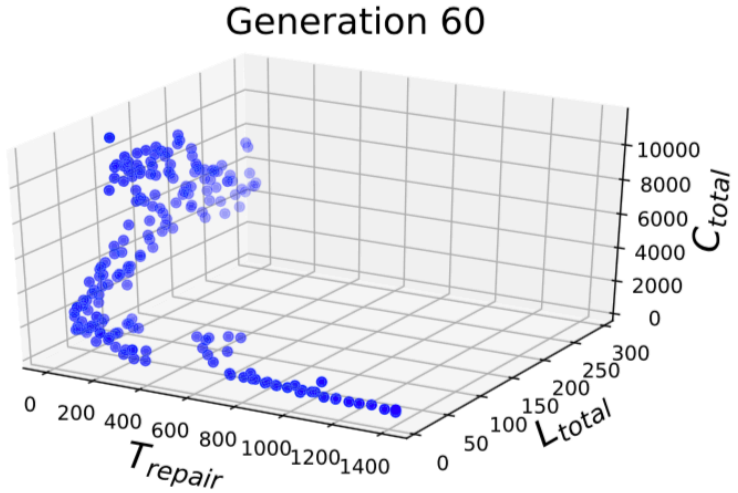

C_{total} (Cost)
min
weighted
Cost units (c)
0
500
1000
1500
0
50
100
150
200
250
Generations

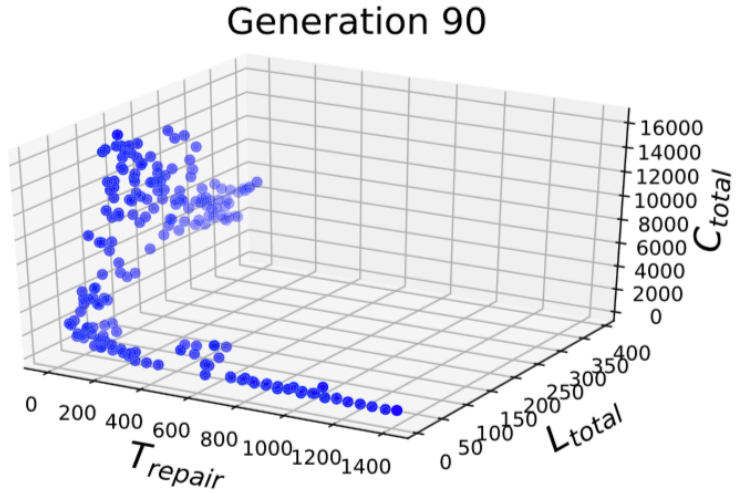

L_{total} (Latency)
min
weighted
Time units (t)
0
25
50
75
0
50
100
150
200
250
Generations


T^{repair} (Repair time)
min
weighted
Time units (t)
0
500
1000
0
50
100
150
200
250
Generations


Generation 0
T_{repair}
L_{total}
C_{total}


Generation 30
T_{repair}
L_{total}
C_{total}


Generation 60
T_{repair}
L_{total}
C_{total}


Generation 90
T_{repair}
L_{total}
C_{total}

Conclusiones

- Los resultados muestran que NSGA-II obtiene valores objetivo en tan solo 200 generaciones con una población de 200 individuos.
- Las soluciones por NSGA-II son siempre mejores que por nuestra aproximación voraz. La propuesta evolutiva obtiene un ratio medio de mejora de 4.09 (309%) en comparación con los resultados de la aproximación voraz.
- Posibilidad de una implementación real: VMWare, OpenNebula, HashiCorp Nomad, etc.

Futuro trabajo

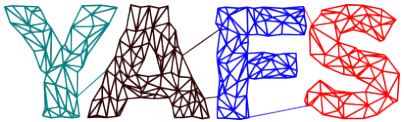
- NSGA-II con otros algoritmos evolutivos como: Firefly o Bat.
- NSGA-II sobre otras tecnologías cloud: serverless, fog computing,..

GOBIERNO DE ESPAÑA · MINISTERIO DE ECONOMÍA, INDUSTRIA Y COMPETITIVIDAD · AGENCIA ESTATAL DE INVESTIGACIÓN
FEDER Fondo Europeo de Desarrollo Regional
UNIÓN EUROPEA
"Una manera de hacer Europa"

Energy and performance optimization for resource and data management in Cloud of Things with Semantic Networks. TIN2017-88547-P (MINECO/AEI/FEDER, UE).

http://ordcot.uib.es/

- *Carlos Guerrero, Isaac Lera, Carlos Juiz,* Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures*, Future Generation Computer Systems, Volume 97, 2019, Pages 131-144, ISSN 0167-739X,*
- *Isaac Lera and Carlos Guerrero and Carlos Juiz* YAFS: A simulator for IoT scenarios in fog computing. *IEEE Access. vol. 7, no. 1, pp. 91745-91758 (2019)*
- *…*



Agradecer vuestra atención

Carlos Guerrero, **Isaac Lera**, Carlos Juiz
Department of Computer Science,
University of Balearic Islands, Spain
Carlos.guerrero@uib.es || isaac.lera@uib.es

http://ordcot.uib.es/