A parallel hybrid evolutionary algorithm for the optimization of broker virtual machines subletting in cloud systems

Santiago Iturriaga, Sergio Nesmachnow*
Universidad de la República, Uruguay
sergion@fing.edu.uy

Bernabé Dorronsoro, El-Ghazali Talbi
University of Lille, France

Pascal Bouvry
University of Luxembourg, Luxembourg

Contents

1. Introduction
2. The Virtual Machine Mapping Problem (VMMP)
3. Literature review
4. A parallel hybrid evolutionary algorithm for the VMPP
5. Experimental analysis
6. Conclusions and future work

Introduction

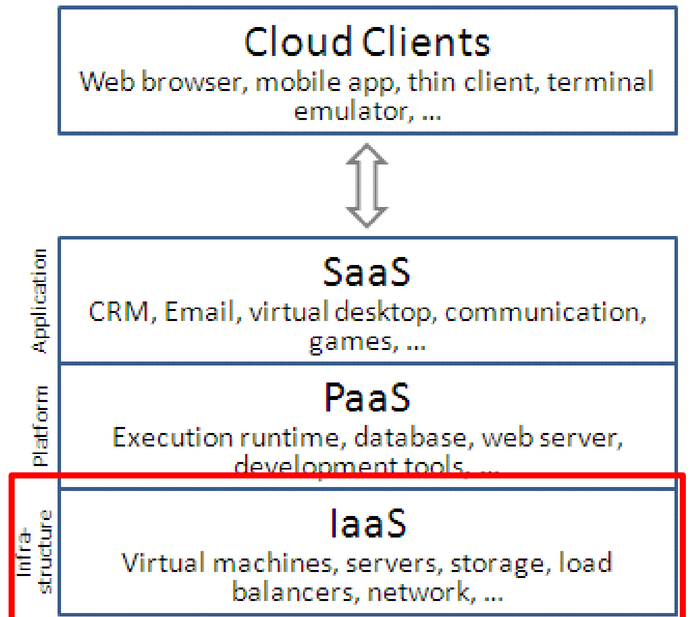
Cloud computing

- Emerged as a powerful computing paradigm due to elasticity, flexibility, and large computational power
- Extends the concept of utility computing, coined in the late 1990s
  - Computing resources as on-demand services
- Offers many services (hardware, software, networking) in several levels

Introduction

Cloud computing

- Emerged as a powerful computing paradigm due to elasticity, flexibility, and large computational power
- Extends the concept of utility computing, coined in the late 1990s
  - Computing resources as on-demand services
- Offers many services (hardware, software, networking) in several levels
- We focus on the Infrastructure as a Service (IaaS) paradigm
  - Offers computing and storage services
  - Based on virtual machines (VMs)



Introduction

IaaS and cloud brokering

IaaS and cloud brokering

- IaaS providers offer two options for hardware:
  - On-demand access (more expensive)
  - Booked instances (cheaper, 12 to 24 months required)

Introduction

IaaS and cloud brokering

- IaaS providers offer two options for hardware:
  - On-demand access (more expensive)
  - Booked instances (cheaper, 12 to 24 months required)
- New agent: the *broker*
  - owns a set of VMs (*reserved instances, RI*) with different features
  - sublets on-demand resources at cheaper prices than the customer would get from a cloud provider
  - if not enough VMs for customer requests without violating SLA, the broker buys on-demand VMs to satisfy the users, and his profit is reduced (he pays more than what he charges the customer for that VM)

Introduction

IaaS and cloud brokering

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- IaaS providers offer two options for hardware:
  - On-demand access (more expensive)
  - Booked instances (cheaper, 12 to 24 months required)
- New agent: the *broker*
  - owns a set of VMs (*reserved instances, RI*) with different features
  - sublets on-demand resources at cheaper prices than the customer would get from a cloud provider
  - if not enough VMs for customer requests without violating SLA, the broker buys on-demand VMs to satisfy the users, and his profit is reduced (he pays more than what he charges the customer for that VM)

This work presents a parallel hybrid evolutionary algorithm for allocating the customers' VM requests into the available RIs from the broker, maximizing the profit

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$

The Virtual Machine Mapping Problem

Formulation

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, $m << n$, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$

The Virtual Machine Mapping Problem

Formulation

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, $m << n$, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*, with $C(b_j) << COD(b_j)$

The Virtual Machine Mapping Problem

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Formulation

- A set of *VM requests* $\{v_1,\dots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\dots,b_m\}$, m << n, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*, with $C(b_j) << COD(b_j)$
- A *pricing function* $p(b_j)$ the broker charges the customers for RI of type b_j.
  - To attract customers p(bj) < COD(bj). If the cheapest RI for a requested VM is not available, the broker can assign a RI of higher capacity, but charging the lower price

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, $m << n$, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*, with $C(b_j) << COD(b_j)$
- A *pricing function* $p(b_j)$ the broker charges the customers for RI of type b_j.
  - To attract customers p(bj) < COD(bj). If the cheapest RI for a requested VM is not available, the broker can assign a RI of higher capacity, but charging the lower price
- Objective: maximize the broker profit

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, m << n, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*, with $C(b_j) << COD(b_j)$
- A *pricing function* $p(b_j)$ the broker charges the customers for RI of type b_j.
  - To attract customers p(bj) < COD(bj). If the cheapest RI for a requested VM is not available, the broker can assign a RI of higher capacity, but charging the lower price
- Objective: maximize the broker profit

$$\max \sum_{j=1}^{j=m} \left(\sum_{i:f(v_i)=b_j} (p(BF(v_i)) - C(b_j)) \times T(v_i) \right) + \sum_{h:ST(v_h)>D(v_h)} (p(BF(v_h)) - COD(BF(v_h))) \times T(v_h)$$

subject to $M(v_i) \leq M(b_j),\ P(v_i) \leq P(b_j)$
$S(v_i) \leq S(b_j),\ nc(v_i) \leq nc(b_j)$

where the $BF(v_k)$ function gives the less expensive instance capable of executing the request v_k

The Virtual Machine Mapping Problem

Formulation

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, m << n, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*,

Profit for RI booked by customers

- A *pricing fu* r charges the customers for RI of type b_j.
  - To attract customers p(bj) < COD(bj). If the cheapest RI for a requested VM is not available, the broker can assign a RI of higher capacity, but charging the lower price
- Objective: maximize the broker profit

$$\max \boxed{\sum_{j=1}^{j=m}\left(\sum_{i:f(v_i)=b_j}(p(BF(v_i))-C(b_j))\times T(v_i)\right)} + \sum_{h:ST(v_h)>D(v_h)}(p(BF(v_h))-COD(BF(v_h)))\times T(v_h)$$

subject to $M(v_i)\le M(b_j),\ P(v_i)\le P(b_j)$
$S(v_i)\le S(b_j),\ nc(v_i)\le nc(b_j)$

where the $BF(v_k)$ function gives the less expensive instance capable of executing the request v_k

The Virtual Machine Mapping Problem

Formulation

- A set of *VM requests* $\{v_1,\ldots,v_n\}$ with time $T(v_i)$ and hardware demands:
  - Processor $P(v_i)$, memory $M(v_i)$, storage $S(v_i)$, and number of cores $nc(v_i)$
- VM requests arrive in batches and must start executing before a *deadline* $D(v_i)$
- A *set of RI* $\{b_1,\ldots,b_m\}$, m << n, pre-booked by the broker, with specific features:
  - Processor $P(b_j)$, memory $M(b_j)$, and storage $S(b_j)$, according to a list of types $t(b_j)$
- A cost function C for *pre-booked RI*, and a cost function COD for *on-demand instances*, w
- A *pricing fu* r charges the cust
  - To attract customers p(bj) < COD(bj). If the cheapest RI for a requested VM is not available, the broker can assign a RI of higher capacity, but charging the lower price
- Objective: maximize the broker profit

Profit for RI booked by customers

Cost for avoiding the deadline constraints

$$\max \boxed{\sum_{j=1}^{j=m}\left(\sum_{i:f(v_i)=b_j}(p(BF(v_i))-C(b_j))\times T(v_i)\right)} + \boxed{\sum_{h:ST(v_h)>D(v_h)}(p(BF(v_h))-COD(BF(v_h)))\times T(v_h)}$$

subject to $M(v_i)\leq M(b_j),\ P(v_i)\leq P(b_j)$
$S(v_i)\leq S(b_j),\ nc(v_i)\leq nc(b_j)$

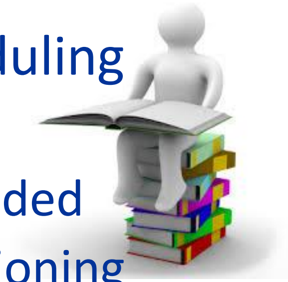
where the $BF(v_k)$ function gives the less expensive instance capable of executing the request v_k

Related work

Literature review

Related work

Literature review



- Calheiros & Buyya (2012): *cloud bursting* for scheduling applications in private resources
  - Enhance local schedulers: use VMs from public clouds when needed
  - Similar to our approach. We do not tackle the resource provisioning problem (our broker always work with VMs from the public cloud)

Related work

Literature review

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Calheiros & Buyya (2012): *cloud bursting* for scheduling applications in private resources
  - Enhance local schedulers: use VMs from public clouds when needed
  - Similar to our approach. We do not tackle the resource provisioning problem (our broker always work with VMs from the public cloud)
- Wu et al. (2008): encourage customers to provide realistic resource utilization, with price reduction rewards
  - Forecast required resources, minimizing underutilization/overbooking
  - Benefits the customer too: service at a low price
  - Applied by Rogers & Cliff (2012) for cloud broker subletting, using the info by the customers to decide buying more resources or not

Related work

Literature review

- Calheiros & Buyya (2012): *cloud bursting* for scheduling applications in private resources
  - Enhance local schedulers: use VMs from public clouds when needed
  - Similar to our approach. We do not tackle the resource provisioning problem (our broker always work with VMs from the public cloud)
- Wu et al. (2008): encourage customers to provide realistic resource utilization, with price reduction rewards
  - Forecast required resources, minimizing underutilization/overbooking
  - Benefits the customer too: service at a low price
  - Applied by Rogers & Cliff (2012) for cloud broker subletting, using the info by the customers to decide buying more resources or not

Our approach: how the broker can manage VMs for maximizing profit and QoS, by using on-demand instances to fulfill the needs of users that cannot be satisfied with current resources, despite the money loss

Proposed scheduler

Scheduling approach

Proposed scheduler

Scheduling approach

- Dynamic approach: based on rescheduling
  - The scheduler executes periodically (reschedule time T_R), or when a pre-booked instance is available

Proposed scheduler

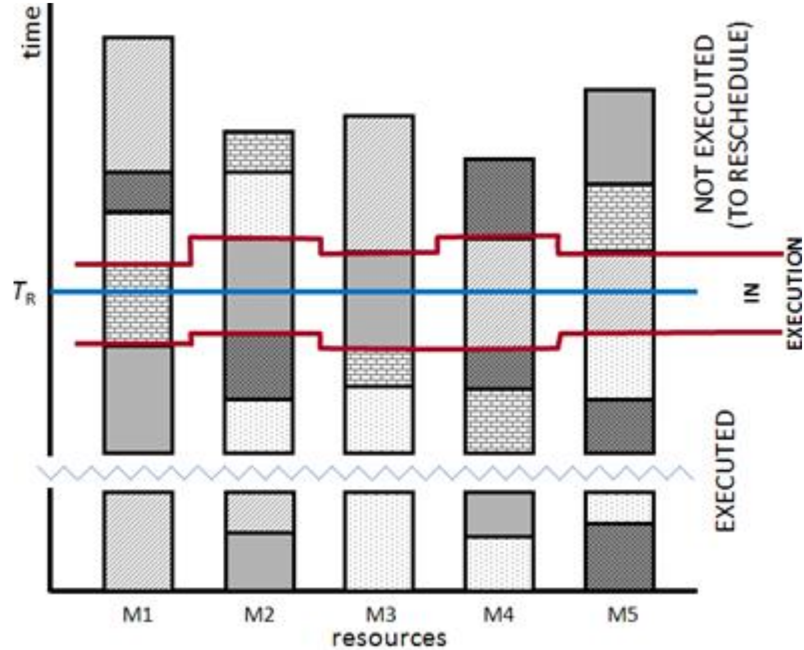
Scheduling approach

- Dynamic approach: based on rescheduling
  - The scheduler executes periodically (reschedule time T_R), or when a pre-booked instance is available
- Rescheduling strategy: find a new schedule for executing the incoming requests (in each new batch) and those submitted requests that have not been executed yet

Proposed scheduler

Scheduling approach

- Dynamic approach: based on rescheduling
  - The scheduler executes periodically (reschedule time T_R), or when a pre-booked instance is available
- Rescheduling strategy: find a new schedule for executing the incoming requests (in each new batch) and those submitted requests that have not been executed yet
  - In each rescheduling, the cost takes into account the remaining time of those VM requests already in execution at time T_R in each RI
  - To model this situation, at time T_R each pre-booked instance has an available start time AS(b_i)



Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

- EA that uses a SA operator for exploiting promising solutions

Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- EA that uses a SA operator for exploiting promising solutions
  - *Encoding*: fixed-size VM-oriented encoding to represent schedules
    - allows an efficient implementation of the evolutionary operators

Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

- EA that uses a SA operator for exploiting promising solutions
  - *Encoding*: fixed-size VM-oriented encoding to represent schedules
    - allows an efficient implementation of the evolutionary operators
  - *Initialization*: randomized Cheapest Instance (rCI) heuristic,
    - randomly assigns VM requests to the cheapest RI that fulfill the requirements

Proposed scheduler

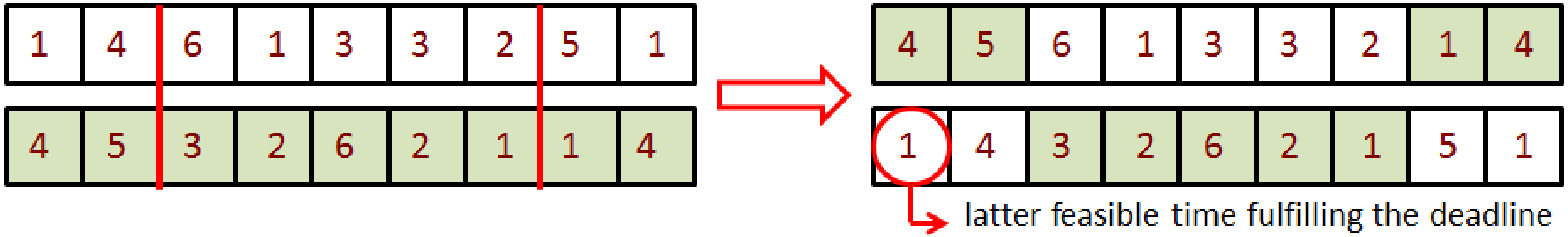
A parallel hybrid evolutionary algorithm (EA+SA)

- EA that uses a SA operator for exploiting promising solutions
  - *Encoding*: fixed-size VM-oriented encoding to represent schedules
    - allows an efficient implementation of the evolutionary operators
  - *Initialization*: randomized Cheapest Instance (rCI) heuristic,
    - randomly assigns VM requests to the cheapest RI that fulfill the requirements
  - *Evolution*: tournament selection, ($\mu+\lambda$) evolution model

Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

- EA that uses a SA operator for exploiting promising solutions
  - *Encoding*: fixed-size VM-oriented encoding to represent schedules
    - allows an efficient implementation of the evolutionary operators
  - *Initialization*: randomized Cheapest Instance (rCI) heuristic,
    - randomly assigns VM requests to the cheapest RI that fulfill the requirements
  - *Evolution*: tournament selection, $(\mu+\lambda)$ evolution model
  - *Crossover*: special 2PX, each VM request is scheduled in the new RI at the latter feasible time satisfying the request deadline

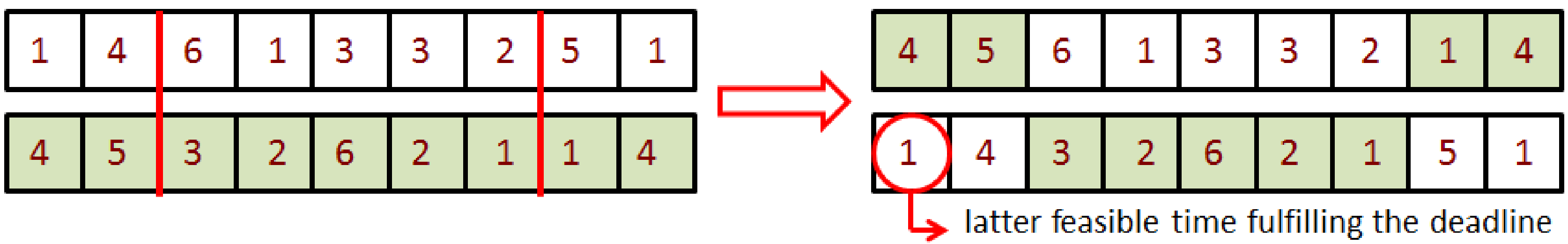


Proposed scheduler

A parallel hybrid evolutionary algorithm (EA+SA)

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- EA that uses a SA operator for exploiting promising solutions
  - *Encoding*: fixed-size VM-oriented encoding to represent schedules
    - allows an efficient implementation of the evolutionary operators
  - *Initialization*: randomized Cheapest Instance (rCI) heuristic,
    - randomly assigns VM requests to the cheapest RI that fulfill the requirements
  - *Evolution*: tournament selection, (μ+λ) evolution model
  - *Crossover*: special 2PX, each VM request is scheduled in the new RI at the latter feasible time satisfying the request deadline



  - *Mutation*: with a low probability a VM request is rescheduled to execute on a randomly selected RI on a random position in the scheduling queue
    - If the (rescheduled) starting time satisfies the deadline requirement, the request is rescheduled. Otherwise, the mutation is discarded

Proposed scheduler

The parallel hybrid evolutionary algorithm (EA+SA)

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Proposed scheduler

The parallel hybrid evolutionary algorithm (EA+SA)

- *SA operator*

Proposed scheduler

The parallel hybrid evolutionary algorithm (EA+SA)

- *SA operator*
  1. find the request with the worst profit (v_W) on a randomly selected set of requests. v_W is rescheduled to execute on-demand if the profit improves

Proposed scheduler

The parallel hybrid evolutionary algorithm (EA+SA)

- *SA operator*
  1. find the request with the worst profit (v_W) on a randomly selected set of requests. v_W is rescheduled to execute on-demand if the profit improves
  2. otherwise, randomly select a set of RIs and reschedule v_W on the RI that most improves the profit (at the latter feasible time regarding its deadline)

Proposed scheduler

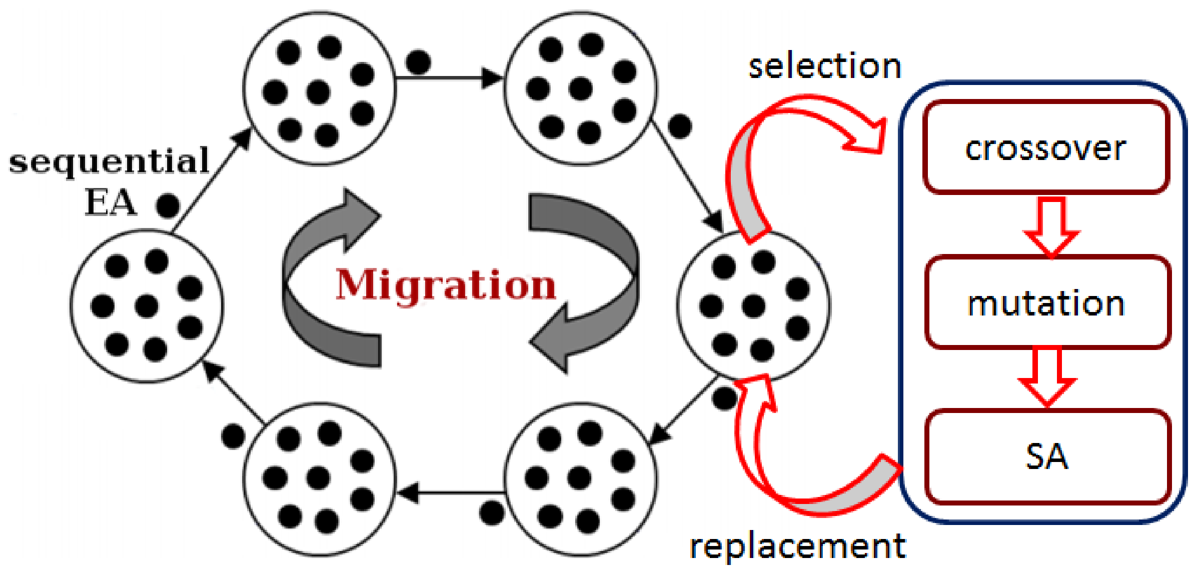
The parallel hybrid evolutionary algorithm (EA+SA)

- *SA operator*
  1. find the request with the worst profit (v_W) on a randomly selected set of requests. v_W is rescheduled to execute on-demand if the profit improves
  2. otherwise, randomly select a set of RIs and reschedule v_W on the RI that most improves the profit (at the latter feasible time regarding its deadline)
- *Parallel model: distributed subpopulations*

Proposed scheduler

The parallel hybrid evolutionary algorithm (EA+SA)

- *SA operator*
  1. find the request with the worst profit (v_W) on a randomly selected set of requests. v_W is rescheduled to execute on-demand if the profit improves
  2. otherwise, randomly select a set of RIs and reschedule v_W on the RI that most improves the profit (at the latter feasible time regarding its deadline)
- *Parallel model: distributed subpopulations*
  - connected on a directed-ring topology. Each subpopulation collaborates with adjacent neighbors subpopulation using a migration operator



Experimental analysis

Methodology

Experimental analysis

Methodology

- 100 VMMP instances: using real data from cloud infrastructures
  - Available at http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP

Experimental analysis

Methodology

- 100 VMMP instances: using real data from cloud infrastructures
  - Available at http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP
- *workloads*: VM requests (memory, storage, processor and cores)
  - batches of 50 to 400 requests per period, durations: 10 to 200 time units

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- 100 VMMP instances: using real data from cloud infrastructures
  - Available at http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP
- *workloads*: VM requests (memory, storage, processor and cores)
  - batches of 50 to 400 requests per period, durations: 10 to 200 time units
- *scenarios*: broker RIs, costs (prebooked/on-demand) and pricing values
  - 10 to 50 RIs, VMs from Amazon and Azure cloud services
  - Pricing function: 20% cheaper than the on-demand price. Reasonable value for attracting users to the service and obtaining interest profit

#	VM id	provider	memory	storage	proc.	nc	price	C	COD
1	m1.small	Amazon	1.7 GB	160 GB	1.0 GHz	1	0.048	0.027	0.06
2	m1.medium	Amazon	3.75 GB	410 GB	2.0 GHz	2	0.096	0.054	0.12
3	A2.medium	Azure	3.5 GB	489 GB	1.6 GHz	2	0.096	0.09	0.12
4	m1.large	Amazon	7.5 GB	850 GB	2.0 GHz	4	0.192	0.108	0.24
5	m2.xlarge	Amazon	17.1 GB	420 GB	3.25 GHz	2	0.192	0.136	0.24
6	A3.large	Azure	7.0 GB	999 GB	1.6 GHz	4	0.328	0.18	0.41
7	c1.xlarge	Amazon	7.0 GB	1690 GB	2.5 GHz	8	0.384	0.316	0.48
8	A4.xlarge	Azure	14.0 GB	2039 GB	1.6 GHz	8	0.464	0.36	0.58

Experimental analysis

Methodology

- 100 VMMP instances: using real data from cloud infrastructures
  - Available at http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP
- *workloads*: VM requests (memory, storage, processor and cores)
  - batches of 50 to 400 requests per period, durations: 10 to 200 time units
- *scenarios*: broker RIs, costs (prebooked/on-demand) and pricing values
  - 10 to 50 RIs, VMs from Amazon and Azure cloud services
  - Pricing function: 20% cheaper than the on-demand price. Reasonable value for attracting users to the service and obtaining interest profit

- Development and execution platform
  - AMD Opteron 6172, 24 core, 2.1GHz, 24GB RAM, Cluster FING: http://www.fing.edu.uy/cluster

Experimental analysis

Parameters setting

Experimental analysis

Parameters setting

- Fixed stopping criterion: 90 seconds of execution time
  - Efficient execution time for on-line cloud planning

Experimental analysis

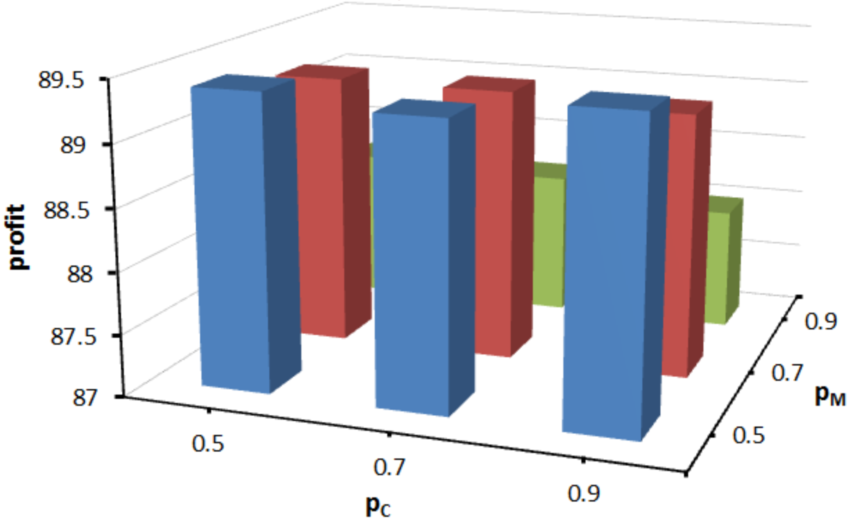
Parameters setting

- Fixed stopping criterion: 90 seconds of execution time
  - Efficient execution time for on-line cloud planning
- Configuration analysis to find the best values for parameters
  - crossover (p_C), mutation (p_M), and SA operator (p_{SA}) probabilities
  - 30 independent executions for each of the 27 parameter combinations
  - Friedman Rank Sum (FRS) test was applied on the results

Experimental analysis

Parameters setting

- Fixed stopping criterion: 90 seconds of execution time
  - Efficient execution time for on-line cloud planning
- Configuration analysis to find the best values for parameters
  - crossover (p_C), mutation (p_M), and SA operator (p_{SA}) probabilities
  - 30 independent executions for each of the 27 parameter combinations
  - Friedman Rank Sum (FRS) test was applied on the results
- Post-hoc analysis of the FRS results: the most accurate schedules were computed when using p_C=0.7, p_M=0.5, and p_{SA}=0.3



Experimental analysis

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance

Experimental analysis

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_H}{makespan_H}$$

Experimental analysis

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_{H}}{makespan_{H}}$$

dimension	**profit improvement**			**makespan gap**		**violations**
	best	*avg.*	*#1*	*CI*	*SRCI*	*avg.*
50×10	133.8%	43.3%	25/25	16.3%	17.7%	2.7%
100×20	47.7%	17.8%	25/25	14.1%	10.0%	2.5%
200×20	46.2%	28.7%	25/25	8.7%	5.4%	9.7%
400×50	63.7%	26.3%	25/25	9.0%	4.5%	5.5%

Experimental analysis

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_{H}}{makespan_{H}}$$

dimension	profit improvement			makespan gap		violations
	best	*avg.*	*#1*	*CI*	*SRCI*	*avg.*
50×10	133.8%	43.3%	$^{25}/_{25}$	16.3%	17.7%	2.7%
100×20	47.7%	17.8%	$^{25}/_{25}$	14.1%	10.0%	2.5%
200×20	46.2%	28.7%	$^{25}/_{25}$	8.7%	5.4%	9.7%
400×50	63.7%	26.3%	$^{25}/_{25}$	9.0%	4.5%	5.5%

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

improved schedules

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_H}{makespan_H}$$

dimension	profit improvement			makespan gap		violations
	best	*avg.*	*#1*	*CI*	*SRCI*	*avg.*
50×10	133.8%	43.3%	$^{25}/_{25}$	16.3%	17.7%	2.7%
100×20	47.7%	17.8%	$^{25}/_{25}$	14.1%	10.0%	2.5%
200×20	46.2%	28.7%	$^{25}/_{25}$	8.7%	5.4%	9.7%
400×50	63.7%	26.3%	$^{25}/_{25}$	9.0%	4.5%	5.5%

Experimental analysis

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

improved schedules

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_H}{makespan_H}$$

dimension	profit improvement			makespan gap		violations
	best	*avg.*	*#1*	*CI*	*SRCI*	*avg.*
50×10	133.8%	43.3%	$^{25}/_{25}$	16.3%	17.7%	2.7%
100×20	47.7%	17.8%	$^{25}/_{25}$	14.1%	10.0%	2.5%
200×20	46.2%	28.7%	$^{25}/_{25}$	8.7%	5.4%	9.7%
400×50	63.7%	26.3%	$^{25}/_{25}$	9.0%	4.5%	5.5%

Results and discussion

- EA+SA compared against greedy list-scheduling heuristics:
  - Cheapest Instance (CI) and Shortest Resource Cheapest Instance (SRCI) from previous work (Nesmachnow et al. 2013)
- 50 independent executions on each VMMP instance
- Evaluate the *gap* metric with respect to each heuristic H:

$$gap_{EA+SA} = \frac{makespan_{EA+SA} - makespan_H}{makespan_H}$$

improved schedules

few deadline violations

dimension	profit improvement			makespan gap		violations
	best	*avg.*	*#1*	*CI*	*SRCI*	*avg.*
50×10	133.8%	43.3%	25/25	16.3%	17.7%	2.7%
100×20	47.7%	17.8%	25/25	14.1%	10.0%	2.5%
200×20	46.2%	28.7%	25/25	8.7%	5.4%	9.7%
400×50	63.7%	26.3%	25/25	9.0%	4.5%	5.5%

Experimental analysis

Parallelism

- Contribution of the parallel model to the results quality

- Contribution of the parallel model to the results quality

dimension	average profit improvement		
	1 deme	*8 demes*	*24 demes*
50×10	$42.89\pm0.39\%$	$43.20\pm0.16\%$	$43.29\pm0.09\%$
100×20	$17.09\pm0.52\%$	$17.60\pm0.31\%$	$17.80\pm0.20\%$
200×20	$24.83\pm1.36\%$	$27.57\pm0.91\%$	$28.71\pm0.75\%$
400×50	$21.34\pm2.04\%$	$24.57\pm1.44\%$	$26.30\pm1.23\%$

Experimental analysis

Parallelism

- Contribution of the parallel model to the results quality

dimension	average profit improvement		
	1 deme	*8 demes*	*24 demes*
50×10	42.89±0.39%	43.20±0.16%	43.29±0.09%
100×20	17.09±0.52%	17.60±0.31%	17.80±0.20%
200×20	24.83±1.36%	27.57±0.91%	28.71±0.75%
400×50	21.34±2.04%	24.57±1.44%	26.30±1.23%

- Using 24 demes account for an additional 5% on the profit results over the sequential search

Conclusions and future work

Parallel EA+SA for the Virtual Machine Mapping Problem

Conclusions and future work

Parallel EA+SA for the Virtual Machine Mapping Problem

- Studied the problem of virtual machines subletting in cloud systems
- A parallel optimization algorithm is proposed for brokering scheduling
  - Combines EA and SA in a weak hybrid method

Conclusions and future work

Parallel EA+SA for the Virtual Machine Mapping Problem

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Studied the problem of virtual machines subletting in cloud systems
- A parallel optimization algorithm is proposed for brokering scheduling
  - Combines EA and SA in a weak hybrid method
- Experimental analysis
  - EA+SA allows tackling large problem instances in reduced execution times
  - Clearly outperform the best existing results in the literature: the broker profit is increased by 18% (average) and up to 133.8% (best)
  - Scalability analysis: profit improves when using parallelism, particularly for the biggest problem instances

Conclusions and future work

Parallel EA+SA for the Virtual Machine Mapping Problem

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

- Studied the problem of virtual machines subletting in cloud systems
- A parallel optimization algorithm is proposed for brokering scheduling
  - Combines EA and SA in a weak hybrid method
- Experimental analysis
  - EA+SA allows tackling large problem instances in reduced execution times
  - Clearly outperform the best existing results in the literature: the broker profit is increased by 18% (average) and up to 133.8% (best)
  - Scalability analysis: profit improves when using parallelism, particularly for the biggest problem instances
- Future work:
  - Further analyze the behavior/dynamics of the new scheduling method
  - Designing an accurate forecasting technique to predict the resources the broker will need in the future

THANKS FOR YOUR ATTENTION

FACULTAD DE INGENIERÍA, UNIVERSIDAD DE LA REPÚBLICA, URUGUAY