Workflow Allocations and Scheduling on IaaS Platforms, from Theory to Practice

Eddy Caron[1], **Frédéric Desprez**[2], Adrian Mureșan[1], Frédéric Suter[3], Kate Keahey[4]

[1]Ecole Normale Supérieure de Lyon, France

[2] INRIA

[3]IN2P3 Computing Center, CNRS, IN2P3

[4]UChicago, Argonne National Laboratory

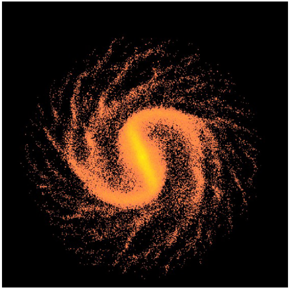
Joint-lab workshop

Outline

Workflows are a common pattern in scientific applications

- applications built on legacy code
- applications built as an aggregate
- use inherent task parallelism
- phenomenons having inherent workflow structure

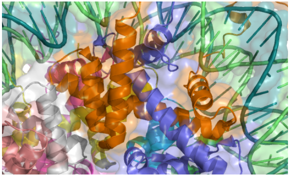
Workflows are omnipresent!



(a) Ramses

(b) Montage

(c) Ocean current modeling



(d) Epigenomics

Figure Workflow application examples

Classic model of resource provisioning

- static allocations in a grid environment
- researchers compete for resources
- researchers tend to **over-provision** and **under-use**
- workflow applications have a **non-constant** resource demand

This is **inefficient**, but can it be improved?
Yes!
How?

- on-demand resources
- automate resource provisioning
- smarter scheduling strategies

Why on-demand resources?

- more efficient resource usage
- eliminate overbooking of resources
- can be easily automated
- unlimited resources *

Our goal

- consider a more general model of workflow apps
- consider on-demand resources and a budget limit
- find a good allocation strategy

Related work

Functional workflows

Bahsi, E.M., Ceyhan, E., Kosar, T.: **Conditional Workflow Management: A Survey and Analysis.** *Scientific Programming 15(4), 283–297 (2007)*

biCPA

Desprez, F., Suter, F.: **A Bi-Criteria Algorithm for Scheduling Parallel Task Graphs on Clusters**. *In: Proc. of the 10th IEEE/ACM Intl. Symposium on Cluster, Cloud and Grid Computing. pp. 243–252 (2010)*

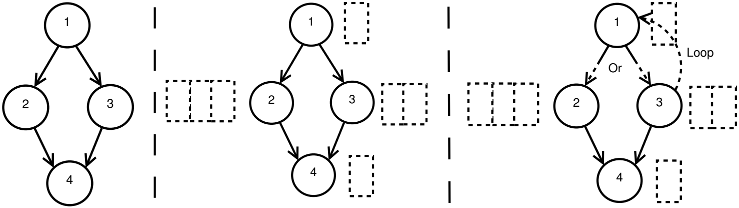
Chemical programming for workflow applications

Fernandez, H., Tedeschi, C., Priol, T.: **A Chemistry Inspired Workflow Management System for Scientific Applications in Clouds**. *In: Proc. of the 7th Intl. Conference on E-Science. pp. 39–46 (2011)*

Pegasus

TMalawski, M., Juve, G., Deelman, E. and Nabrzyski, J..: **Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds**. *24th IEEE/ACM International Conference on Supercomputing (SC12) (2012)*

Outline



(a) Classic (b) PTG (c) Functional

Figure Workflow types

Application model

Non-deterministic (functional) workflows

An application is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where

$\mathcal{V} = \{v_i | i = 1, \ldots, |V|\}$ is a set of vertices

$\mathcal{E} = \{e_{i,j} | (i,j) \in \{1, \ldots, |V|\} \times \{1, \ldots, |V|\}\}$ is a set of edges representing precedence and flow constraints

Vertices

- a computational task [*parallel*, *moldable*]
- an **OR-split** [transitions described by random variables]
- an **OR-join**

Example workflow

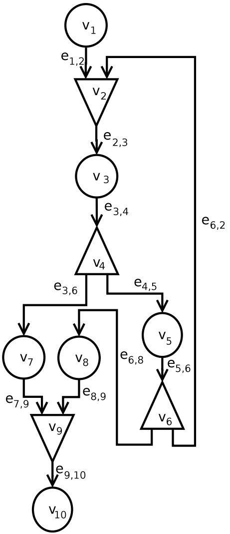


Figure Example workflow

Platform model

A provider of on-demand resources from a catalog:

$\mathcal{C} = \{vm_i = (nCPU_i, cost_i) | i \geq 1\}$

nCPU represents the number of equivalent virtual CPUs

cost represents a monetary cost per running hour (Amazon-like)

communication bounded multi-port model

Makespan

$C = \max_i C(v_i)$ is the global makespan where

$C(v_i)$ is the finish time of task $v_i \in \mathcal{V}$

Cost of a schedule $\mathcal{S}$

$Cost = \sum_{\forall vm_i \in \mathcal{S}} \lceil T_{end_i} - T_{start_i} \rceil \times cost_i$

T_{start_i}, T_{end_i} represent the start and end times of vm_i

$cost_i$ is the catalog cost of virtual resource vm_i

Problem statement

Given

$\mathcal{G}$	a workflow application
$\mathcal{C}$	a provider of resources from the catalog
$\mathcal{B}$	a budget

find a schedule $\mathcal{S}$ such that

$Cost \leq \mathcal{B}$	budget limit is not passed
C	(makespan) is minimized

with a predefined confidence.

Proposed approach

1. Decompose the non-DAG workflow into DAG sub-workflows
2. Distribute the budget to the sub-workflows
3. Determine allocations by adapting an existing allocation approach

Step 1: Decomposing the workflow

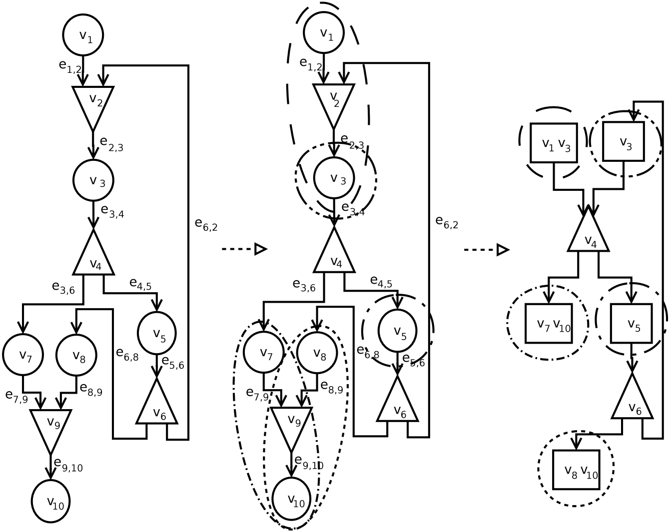


Figure Decomposing a nontrivial workflow

Step 2: Allocating budget

1. Compute the number of executions of each sub-worflow
   - # of transitions of the edge connecting its parent OR-split to its start node
   - Described by a random variable according to a distinct normal distribution + confidence parameter

2. Give each sub-workflow a ratio of the budget proportional to its *work contribution*.

Work contribution of a sub-workflow $\mathcal{G}^i$

- as the sum of the average execution times of its tasks
- average execution time computed over the catalog $\mathcal{C}$
- task speedup model is taken into consideration
- multiple executions of a sub-workflow also considered

Step 3: Determining allocations

Two algorithms based on the bi-CPA algorithm.

Eager algorithm

- one allocation for each task
- good trade-off between makespan and average time-cost area
- fast algorithm
- considers allocation-time cost estimations only

Deferred algorithm

- outputs multiple allocations for each task
- good trade-off between makespan and average time-cost area
- slower algorithm
- one allocation is chosen at scheduling time

Algorithm parameters

T_A^{over}, T_A^{under} average work allocated to tasks

T_{CP} duration of the critical path

B' estimation of the used budget when T_A and T_{CP} meet

- T_A keeps increasing as we increase the allocation of tasks and T_{CP} keeps decreasing so they will eventually meet.
- When they do meet we have a trade-off between the average work in tasks and the makespan.

$p(v_i)$ number of processing units allocated to task v_i

The eager allocation algorithm

for all $v \in \mathcal{V}^i$ **do**
  $Alloc(v) \leftarrow \{\min_{vm_i \in \mathcal{C}} CPU_i\}$
end for
Compute B'
while $T_{CP} > T_A^{over} \cap \sum_{j=1}^{|\mathcal{V}^i|} cost(v_j) \leq B^i$ **do**
  for all $v_i \in$ Critical Path **do**
    Determine $Alloc'(v_i)$ such that $p'(v_i) = p(v_i) + 1$
    $Gain(v_i) \leftarrow \frac{T(v_i, Alloc(v_i))}{p(v_i)} - \frac{T(v_i, Alloc'(v_i))}{p'(v_i)}$
  end for
  Select v such that $Gain(v)$ is maximal
  $Alloc(v) \leftarrow Alloc'(v)$
  Update T_A^{over} and T_{CP}
end while

Algorithm 1: Eager-allocate($\mathcal{G}^i = (\mathcal{V}^i, \mathcal{E}^i), B^i$)

Methodology

- Simulation using SimGrid
- Used 864 synthetic workflows for three types of applications
  - Fast Fourier Transform
  - Strassen matrix multiplication
  - Random workloads
- Used a virtual resource catalog inspired by Amazon EC2
- Used a classic list-scheduler for task mapping
- Measured
  - Cost and makespan after task mapping

Name	#VCPUs	Network performance	Cost / hour
m1.small	1	*moderate*	0.09
m1.med	2	*moderate*	0.18
m1.large	4	*high*	0.36
m1.xlarge	8	*high*	0.72
m2.xlarge	6.5	*moderate*	0.506
m2.2xlarge	13	*high*	1.012
m2.4xlarge	26	*high*	2.024
c1.med	5	*moderate*	0.186
c1.xlarge	20	*high*	0.744
cc1.4xlarge	33.5	10 Gigabit Ethernet	0.186
cc2.8xlarge	88	10 Gigabit Ethernet	0.744

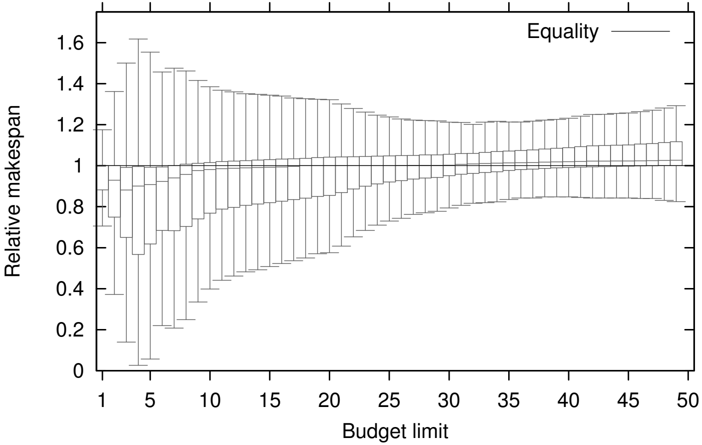


Figure Relative makespan ($\frac{Eager}{Deferred}$) for all workflow applications

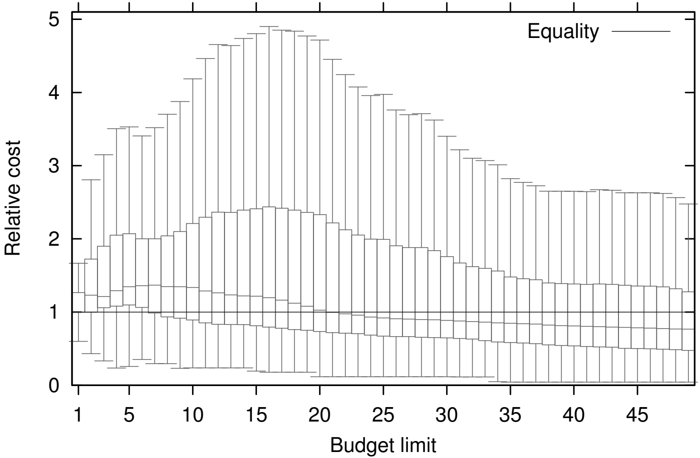


Figure Relative cost ($\frac{Eager}{Deferred}$) for all workflow applications

First conclusions

- Eager is fast but cannot guarantee budget constraint after mapping
- Deferred is slower, but guarantees budget constraint
- After a certain budget they yield to identical allocations
- for small applications and small budgets Deferred should be preferred.
- When the size of the applications increases or the budget limit approaches task parallelism saturation, using Eager is preferable.

Outline

Architecture

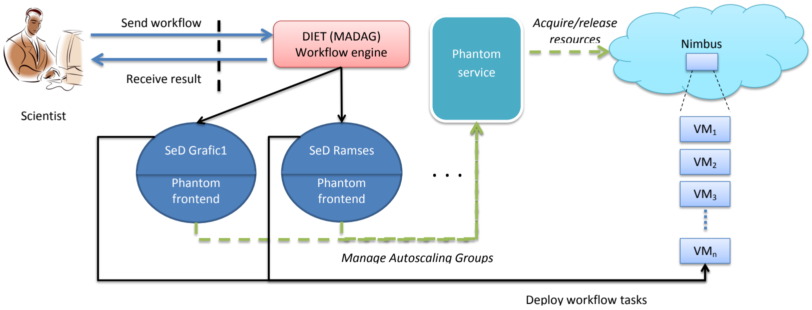


Figure System architecture

Main components

Nimbus

- open-source IaaS provider
- provides low-level resources (VMs)
- compatible with the Amazon EC2 interface
- used a FutureGrid install

Main components

Phantom

- auto-scaling and high availability provider
- high-level resource provider
- subset of the Amazon auto-scale service
- part of the Nimbus platform
- used a FutureGrid install
- still under development

Main components

MADag

- workflow engine
- part of the DIET (Distributed Interactive Engineering Toolkit) software
- one service implementation per task
- each service launches its afferent task
- supports DAG, PTG and functional workflows

Main components

Client

- describes his workflow in xml
- implements the services
- calls the workflow engine
- **no explicit resource management**
- selects the IaaS provider to deploy on

How does it work?

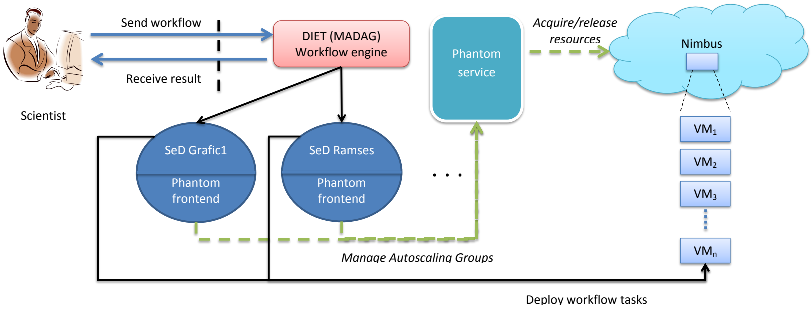


Figure System architecture

RAMSES

- n-body simulations of dark matter interactions
- backbone of galaxy formations
- AMR workflow application
- parallel (MPI) application
- can refine at different zoom levels

RAMSES

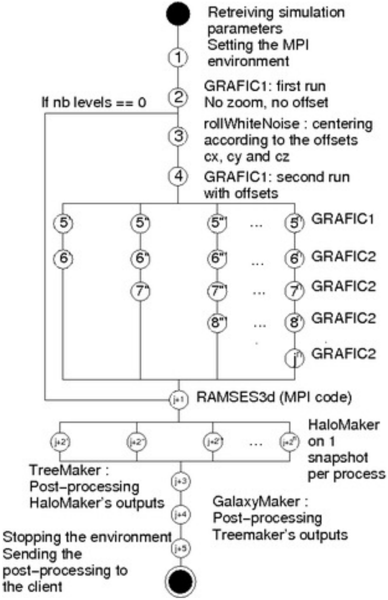


Figure The RAMSES workflow

Methodology

- used a **FutureGrid** Nimbus installation as a testbed
- measured **running time** for **static** and **dynamic** allocations
- estimated **cost** for each allocation
- varied maximum number of used resources

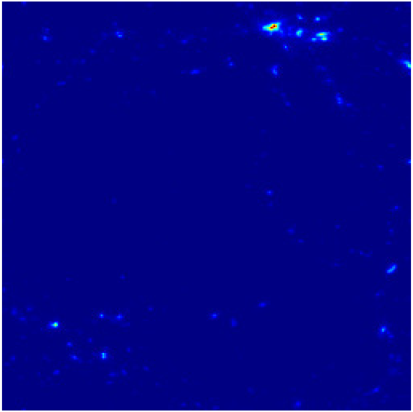


Figure Slice through a $2^8 \times 2^8 \times 2^8$ box simulation

Results

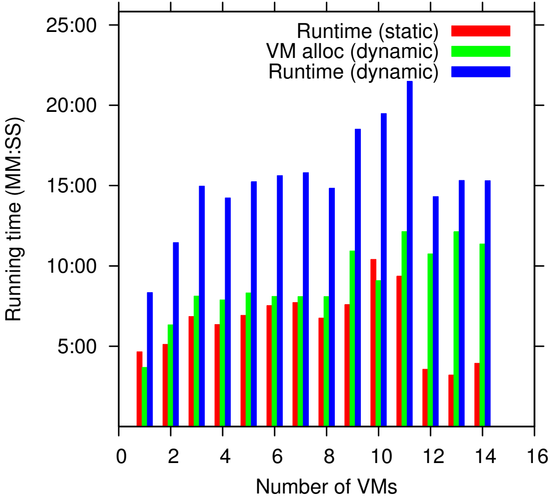


Figure Running times for a $2^6 \times 2^6 \times 2^6$ box simulation

Results

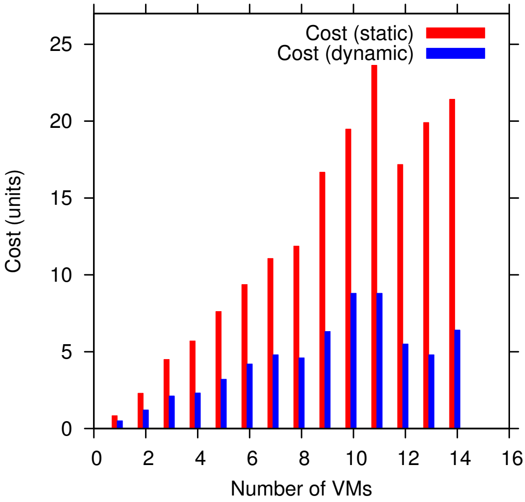


Figure Estimated costs for a $2^6 \times 2^6 \times 2^6$ box simulation

Outline

Conclusions

- proposed two algorithms – Eager and Deferred with each their pro and cons
- on-demand resources can better model workflow usage
- on-demand resources have a VM allocation overhead
- allocation overhead decreases with number of VMs
- for RAMSES, cost is greatly reduced

Perspectives

- preallocate VMs
- spot instances
- smarter scheduling strategy
- determine per application type which is the tipping point
- Compare our algorithms with others

Collaborations

- Continue the collaboration with the Nimbus/FutureGrid teams
- On the algorithms themselves (currently too complicated for an actual implementation)
- Understanding (obtaining models) clouds and virtualized platforms
- going from theoretical algorithms to (accurate) simulations and actual implementation

References

Eddy Caron, Frédéric Desprez, Adrian Muresan and Frédéric Suter. **Budget Constrained Resource Allocation for Non-Deterministic Workflows on a IaaS Cloud.** *12th International Conference on Algorithms and Architectures for Parallel Processing.* (ICA3PP-12), Fukuoka, Japan, September 04 - 07, 2012

Adrian Muresan, Kate Keahey. **Outsourcing computations for galaxy simulations.** *In eXtreme Science and Engineering Discovery Environment 2012 - XSEDE12*, Chicago, Illinois, USA, June 15 - 19 2012. **Poster session.**

Adrian Muresan. **Scheduling and deployment of large-scale applications on Cloud platforms.** *Laboratoire de l'Informatique du Parallélisme (LIP), ENS Lyon, France, Dec. 10th, 2012* **PhD thesis.**