Opencloudware

Journée ARC6 Web & Cloud

Lyon, Mercredi 21 novembre 2012

Christophe Gravier [1], Julien Subercaze [1], Frédérique Laforest [1]

[1] Université Jean Monnet Saint-Étienne and Institut Mines-Télécom, France

Qui sommes-nous ?

- Équipe de recherche SATIN, Laboratoire Télécom Claude Chappe
  - équipe de recherche associée à l'Institut Mines Télécom
- 4 permanents (2PR 1MC 1CR) + 6 doctorants + 1 ingé d'études
- Topics
  - historique : context-awareness, ubiquitous/pervasive computing
  - actuel : fouille, analyse, prédiction pour le Web social et sémantique, sous contrainte de passage à l'échelle et de temps.
- Cas d'applications : Web, cloud, presse, énergie.
- Volonté d'aller jusqu'au prototype :
  `http://demo-satin.telecom-st-etienne.fr`

Agenda

1. Openclouware R&D project
   - Presentation
   - Use case
   - Technical benefits
2. End-user context
   - Context-awareness
   - Illustrative scenario
3. Big (semantic) data
   - Semantic Web for context-awareness
   - Distributed reasoning
   - Cloud privé
4. Conclusion

Agenda

Agenda

Opencloudware project

- Co-funded collaborative R&D project, 18 partners
- Following CompatibleOne, presented this morning.

Opencloudware project

- Co-funded collaborative R&D project, 18 partners
- Following CompatibleOne, presented this morning.
- Objective : Enables to easily build, generate and operate enterprise distributed applications for deployment on any cloud, which includes :
  1. **Complete complex application (vApp) lifecycle management** (Modeling, assembly, and build)
  2. Deployment, operations, re-configurations (PaaS) **at runtime**
  3. On possibly **multi-IaaS**

Opencloudware project

- Co-funded collaborative R&D project, 18 partners
- Following CompatibleOne, presented this morning.
- Objective : Enables to easily build, generate and operate enterprise distributed applications for deployment on any cloud, which includes :
  1. **Complete complex application (vApp) lifecycle management** (Modeling, assembly, and build)
  2. Deployment, operations, re-configurations (PaaS) **at runtime**
  3. On possibly **multi-IaaS**
- 3-year project (jan 2012 – dec 2014)
- Supported by the French FSN (Fonds National pour la Société Numérique)

Partners

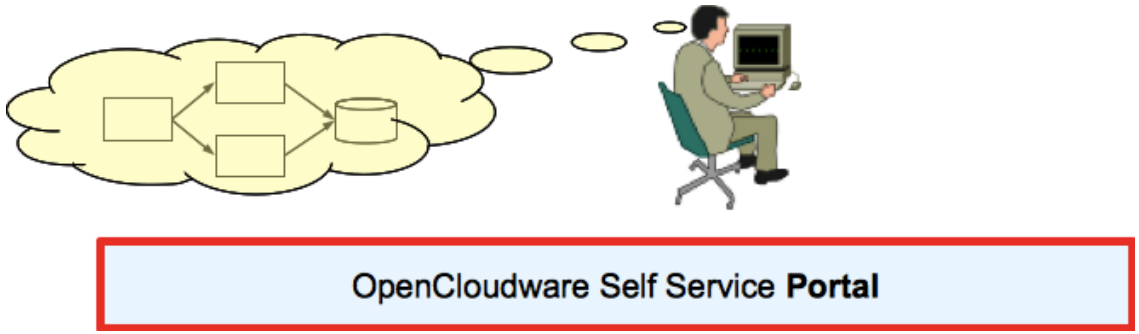
Industry leaders and innovative SMEs / SaaS solution providers

Academic partners

Open source community management

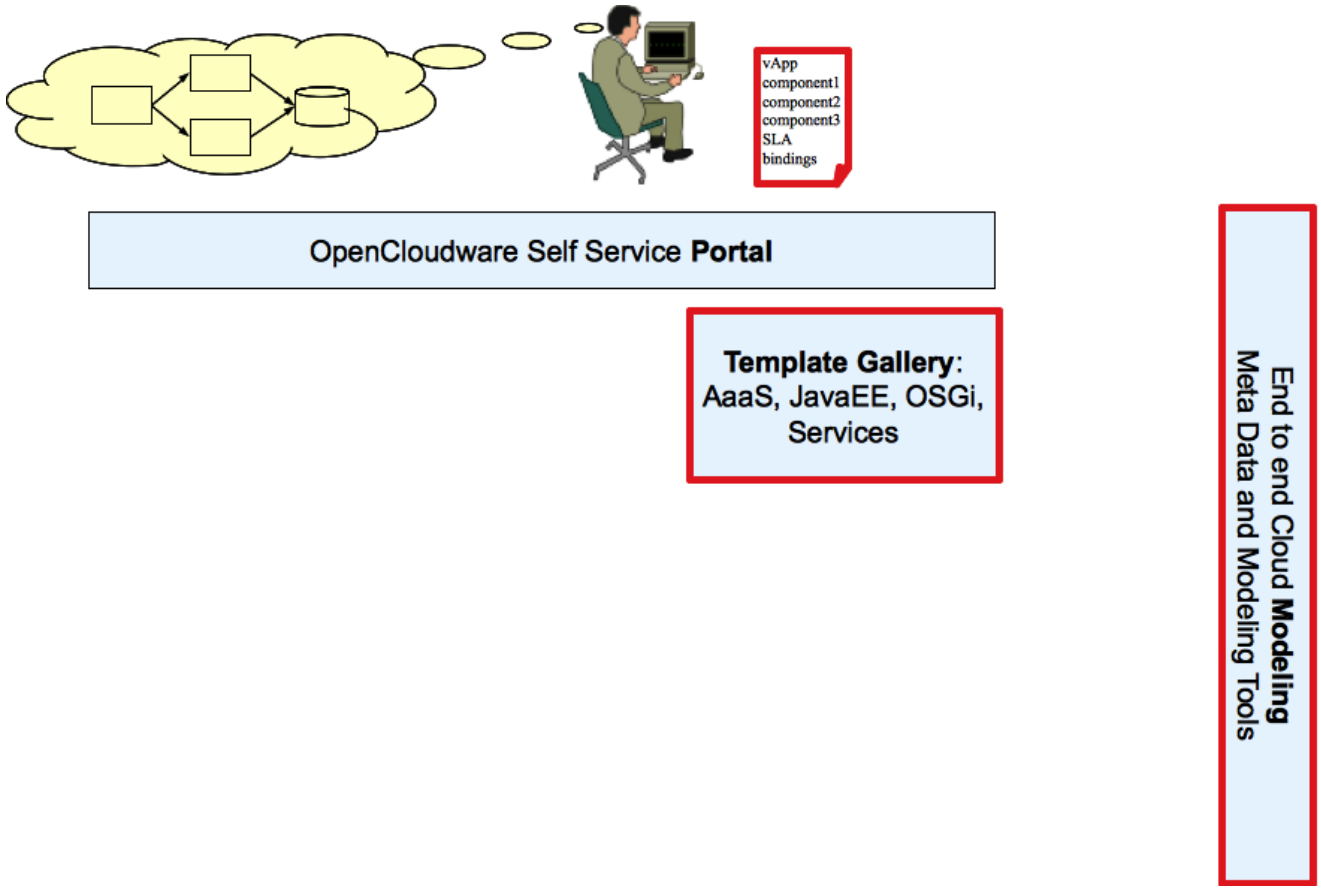
Agenda

Use case



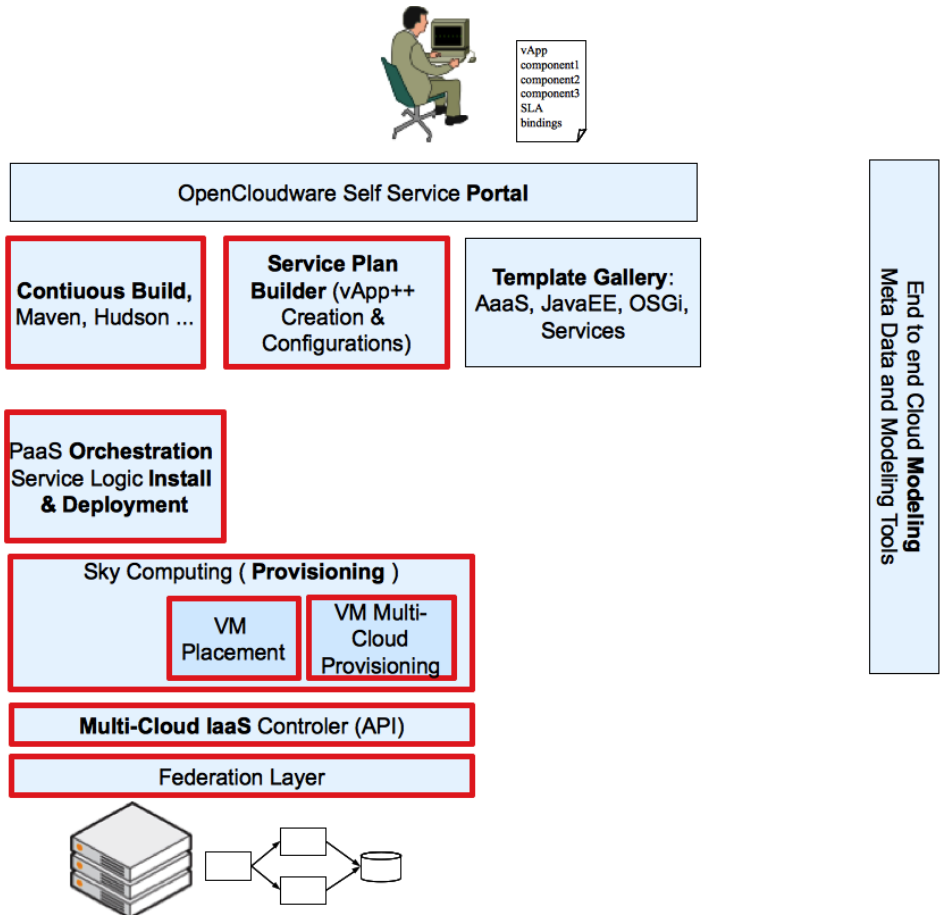
- The user wants to deploy a distributed enterprise application (e.g. LAMP),
- First step : account creation and login.
- Portail = base eXo Portal, authent/règles = base AuthzForce - Thalès - open source dans OW2)

Use case



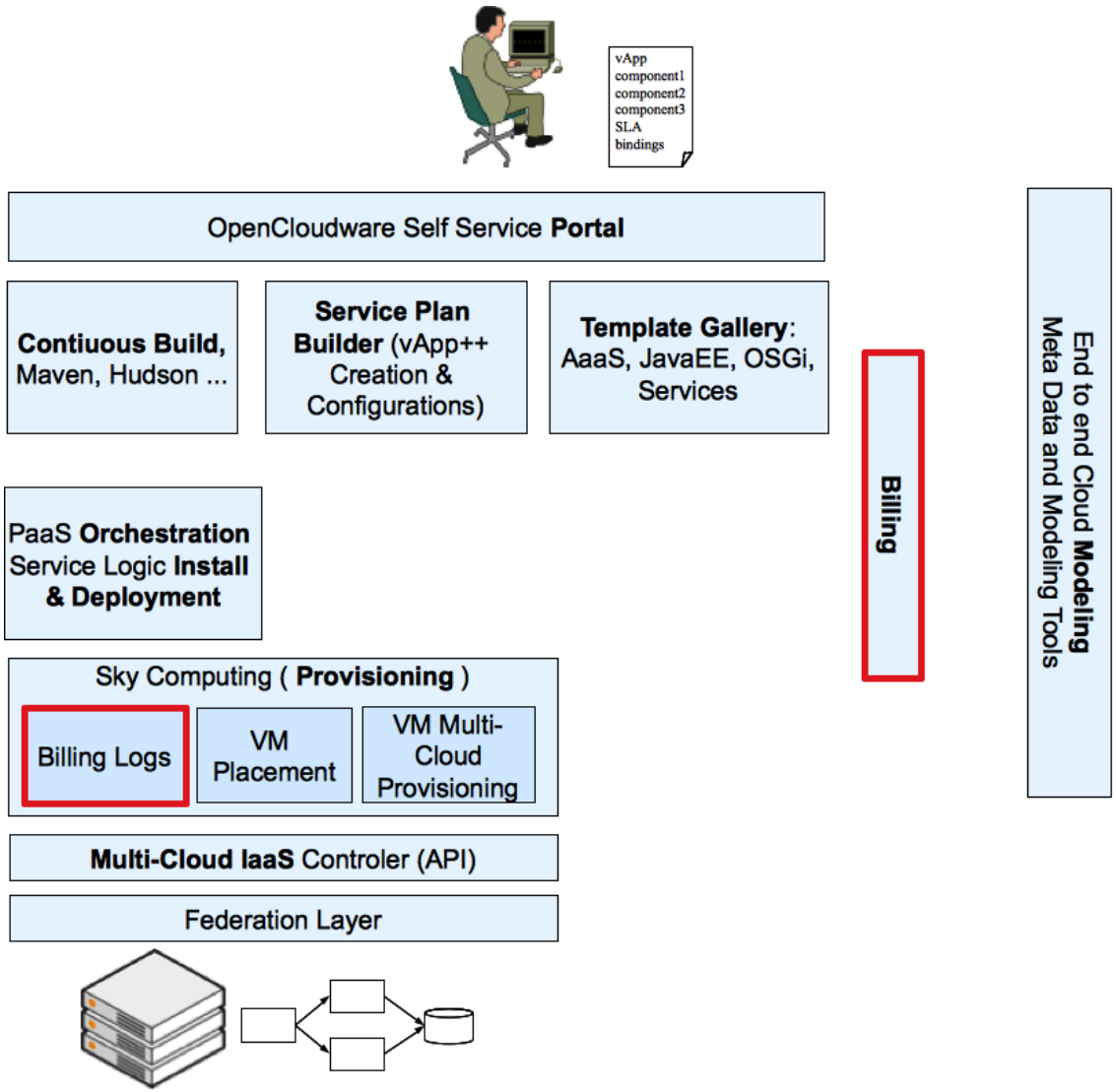
- In the desc., we include vApp, SLAs, elasticity rules.
- Modelization possibly collaborative.
- OpenCloudware develops its own model, as an extension to OVF (WIP).

Use case



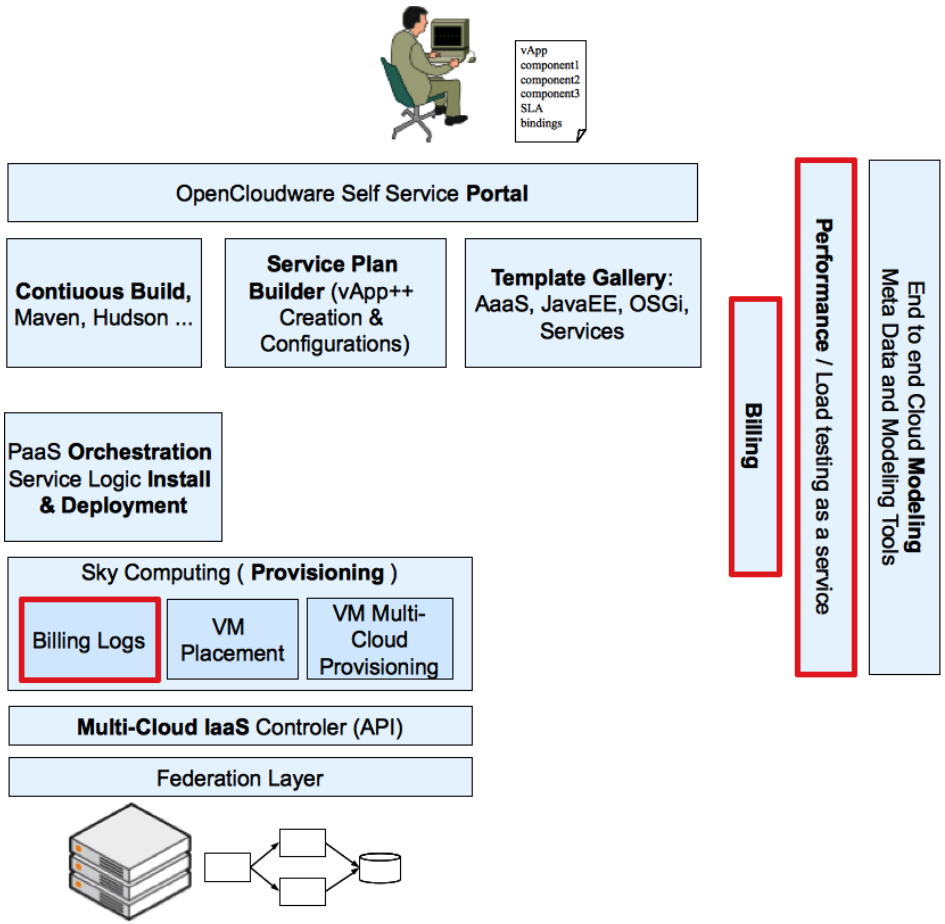
- The vApp is built, and OpenCloudware PaaS instanciates the user vApp.
- Multi IaaS API Controller brought by OW2 Sirocco, implementing DMTF CIMI.

Use case



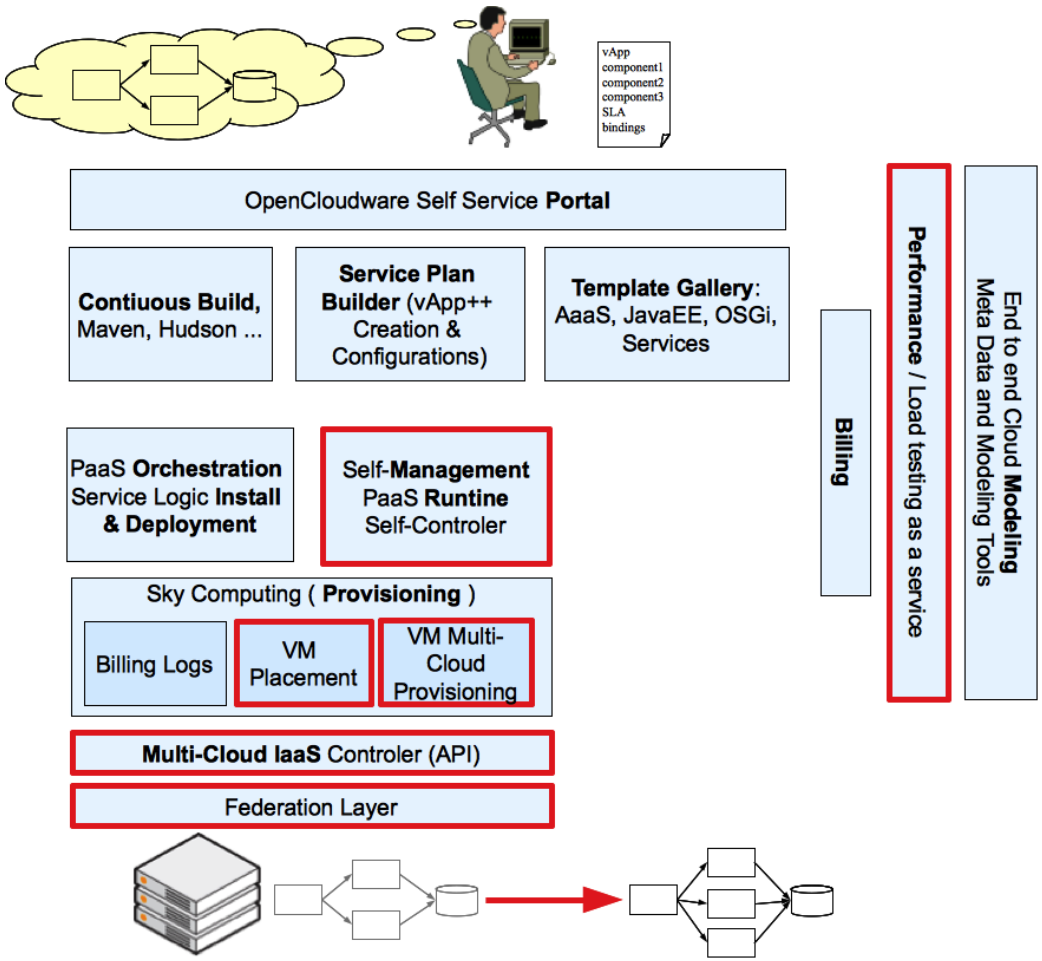
- Development mode, the users can evaluate how much it will cost.

Use case



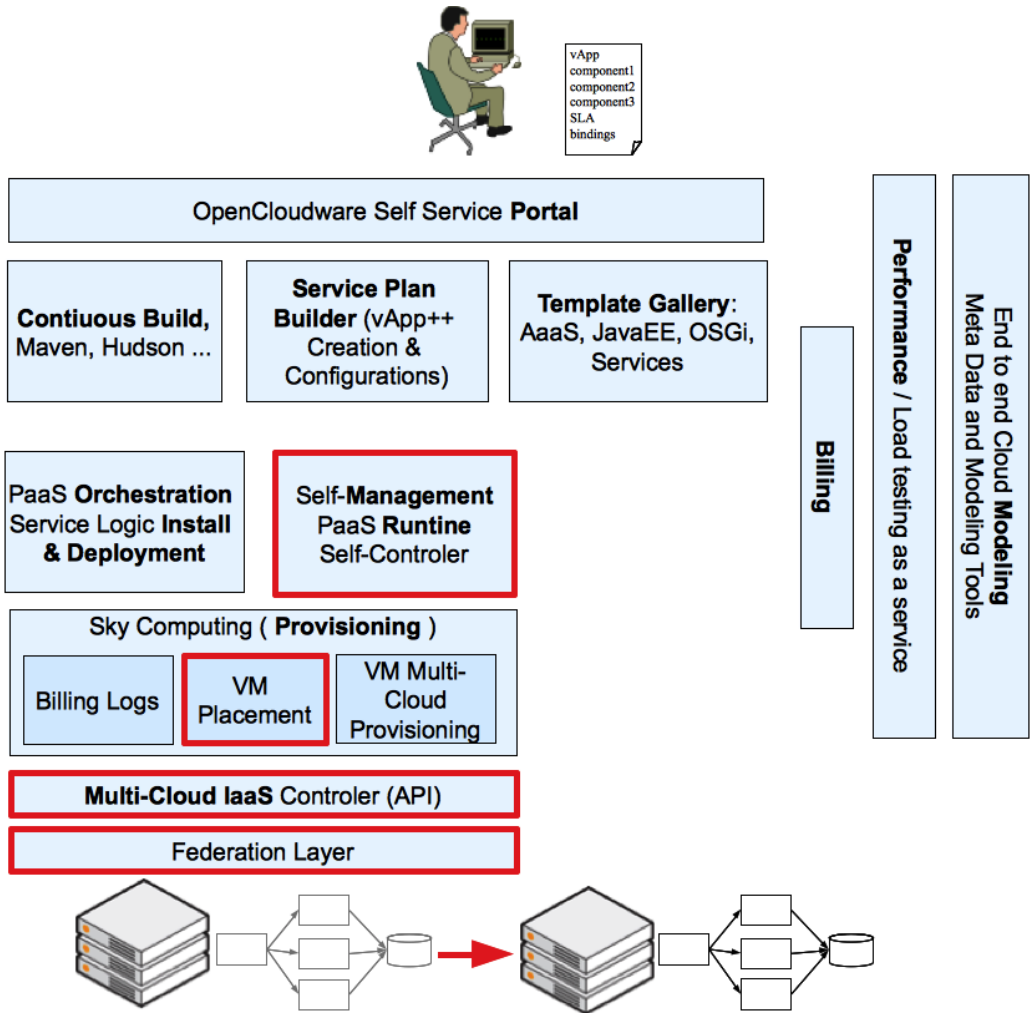
- Benchmarking as a service (CLIF-based, FT) to verify scalability at a given workload

Use case



- Testing's over, the user gets public IP, modifies the vApp description, redeploys.
- The user can verify elasticity using the workbench tool

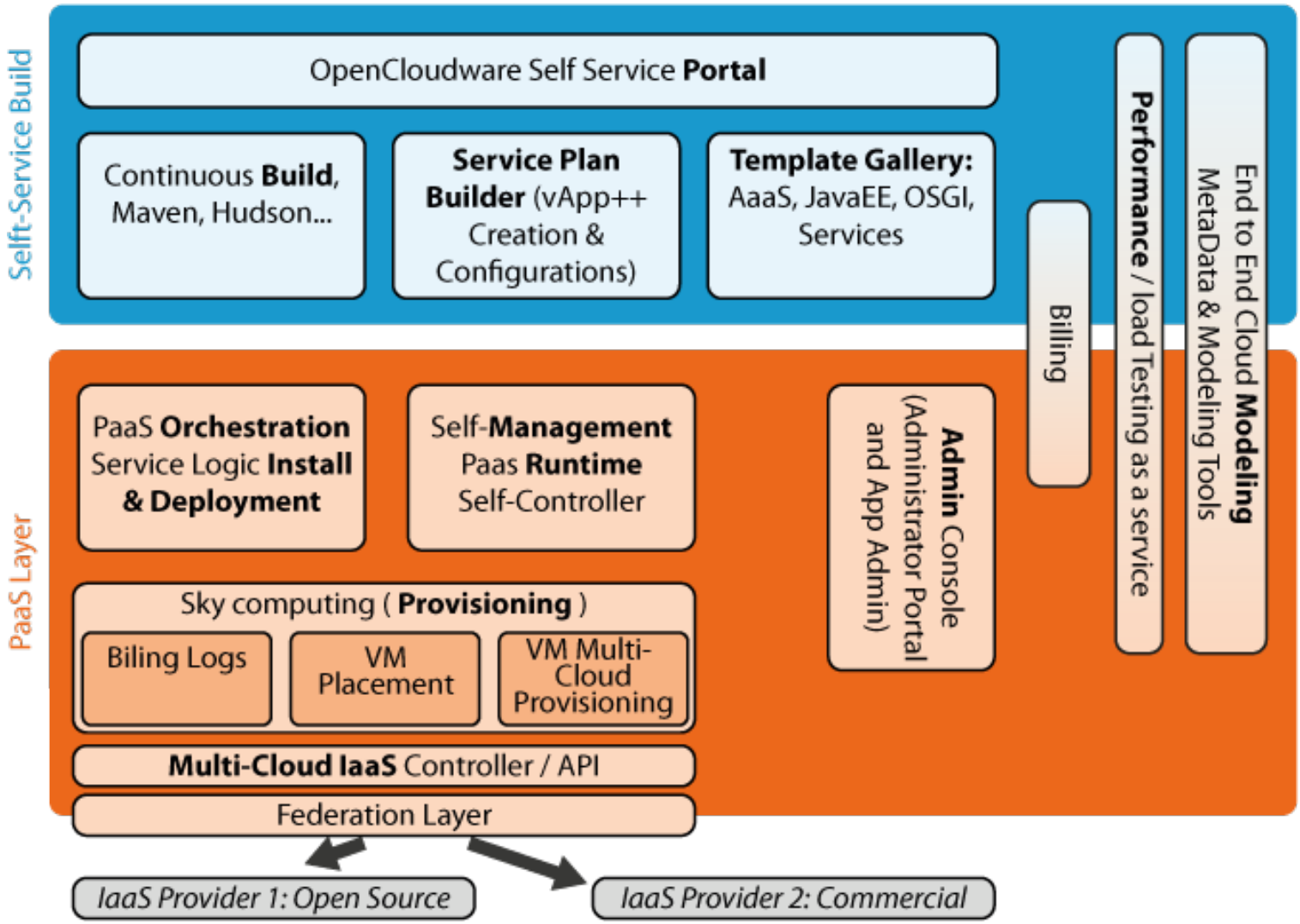
Use case



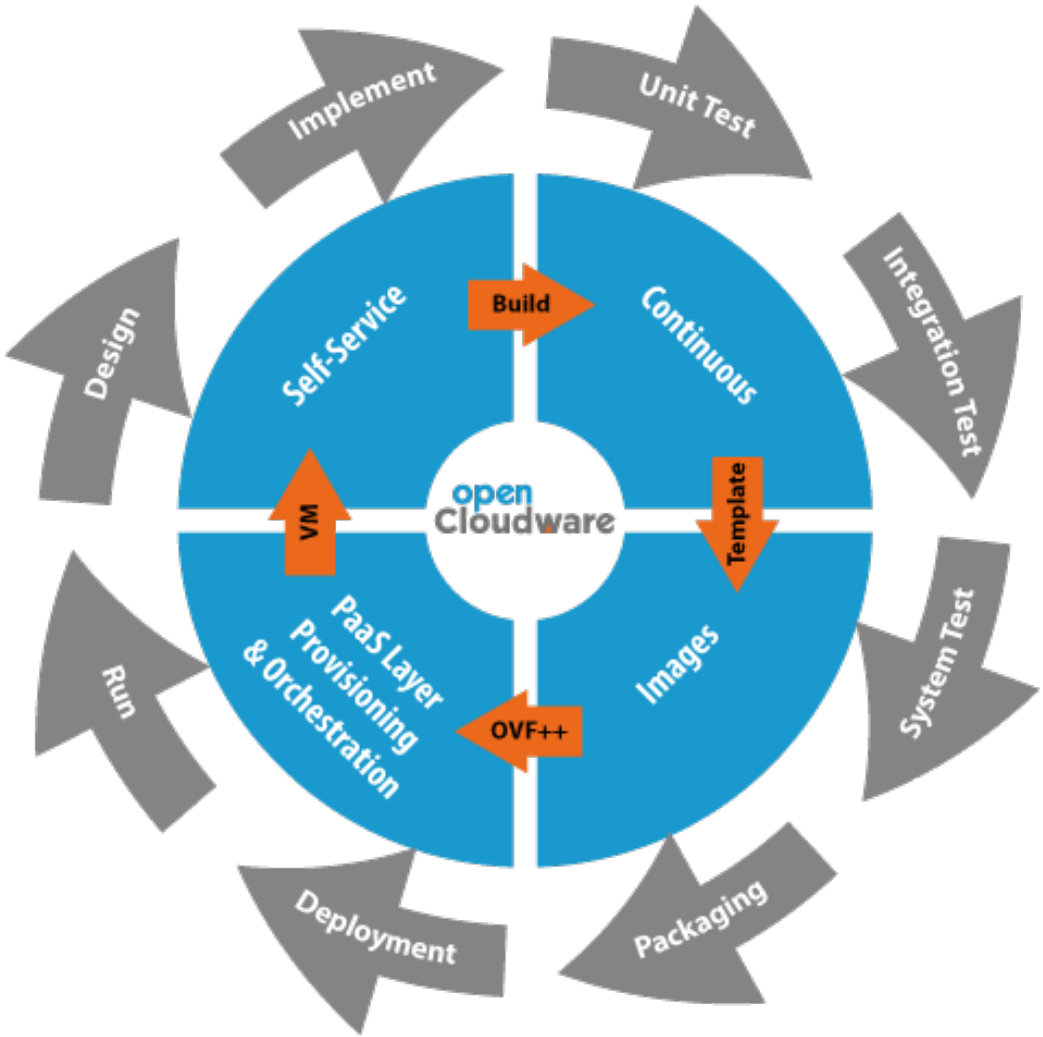
- The application goes to production.
- Opencloudware provides monitoring tools

Agenda

Open source architecture



Lifecycle


Implement
Unit Test
Integration Test
System Test
Packaging
Deployment
Run
Design
Self-Service
Build
Continuous
VM
open
Cloudware
Template
PaaS Layer
Provisioning
& Orchestration
OVF++
Images

Issues addressed

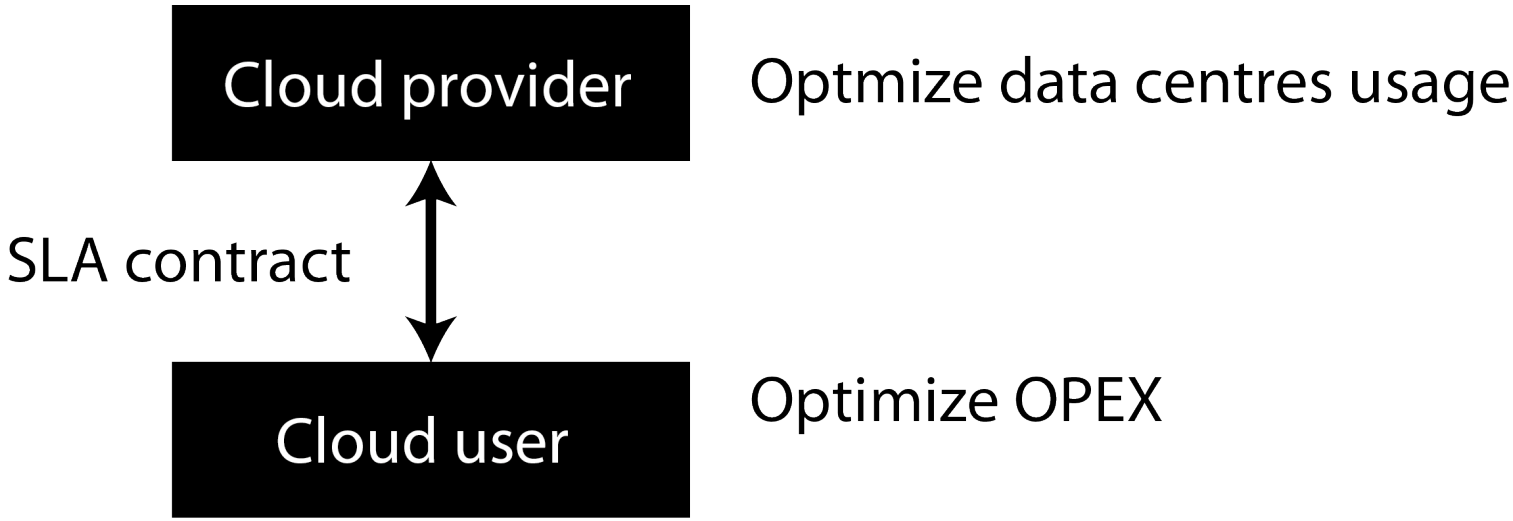
- E2E Modeling : Full stack from OS to applications and configurations incl. SLA, PaaS services and IaaS services
- E2E Automation : Life cycle automation from Dev to Cloud to Cloud orchestration
- Build vApps : Service Factory for visual design and automated generation of multi-tier server templates
- Autonomic management : Dynamic cloud software (VMs) evolutions, allowing elasticity, Green IT optimisation, reliability...
- Multi-IaaS : Portability at the IaaS level, IaaS Agnostic services
- Security : Isolation of applications and security, identity and access control management

Links with other open source cloud projets such as FUI CompatibleOne, FP7 Contrail, . . . (Other identified projects : ANR/FUI SelfXL, ANR MyCloud, ITEA EasiCloud, FP7 4Caast, FUI CoolIT, FUI Energetic, FUI AGOS, ANR OMD2)

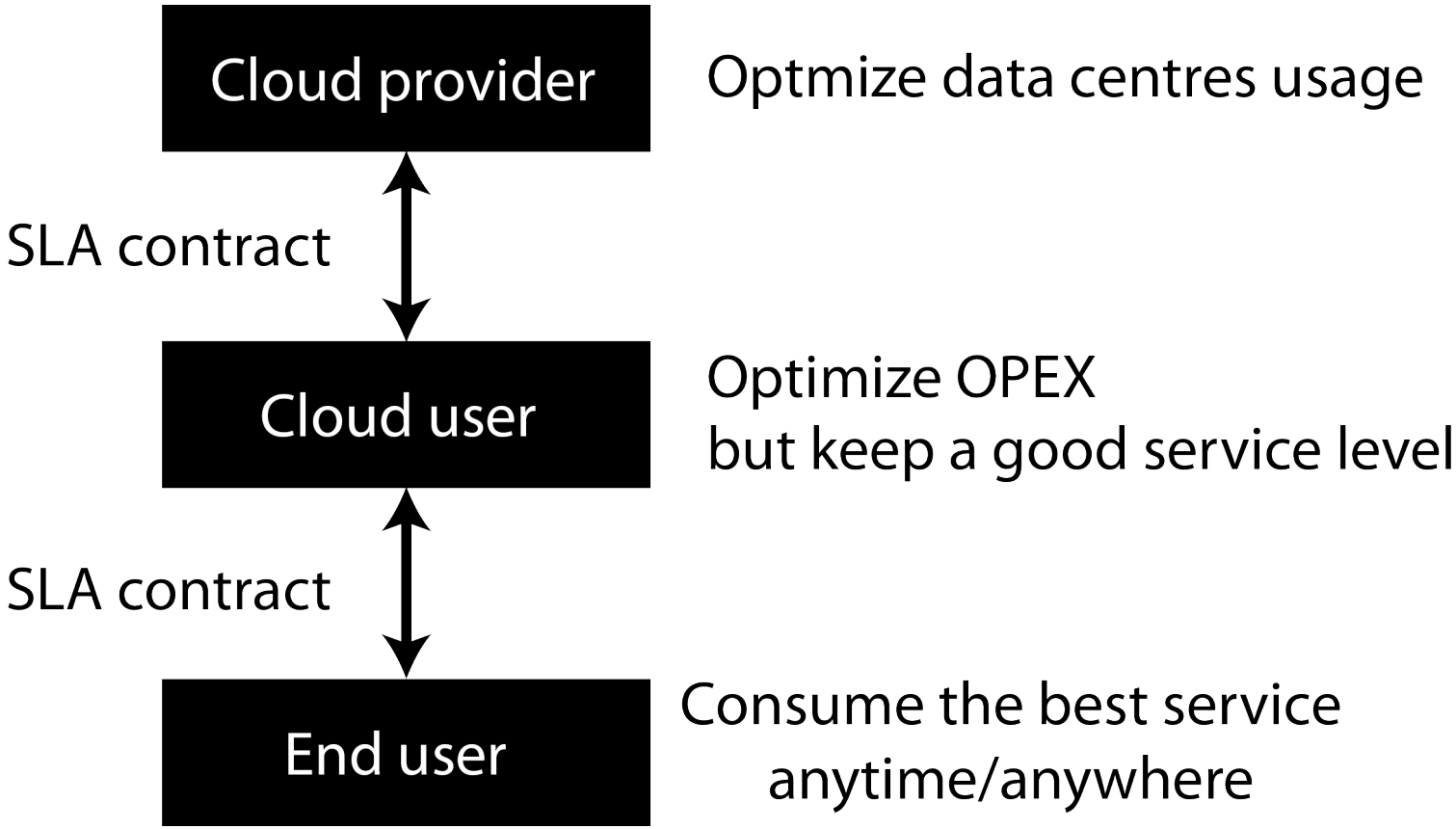
Agenda

Agenda

Two-actors schema



But it's three actors !



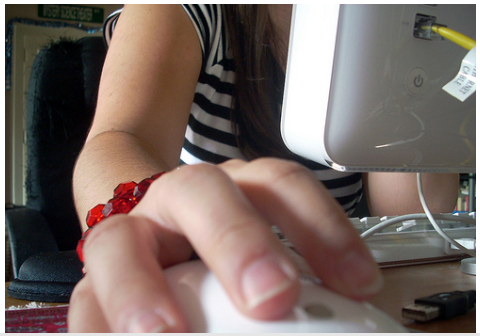
End-user : where, what, using, profile, . . .

End-user : where, what, using, profile, . . .

End-user : where, what, using, profile, . . .

- Cloud elasticity should not be limited to vertical and horizontal sizing @IaaS level : cloud services should adapt their BL to the context of use
  → PaaS provider duty to offer configuration and runtime environment.

End-user : where, what, using, profile, . . .



Is it worth it ? For whose interest ?

- Cloud elasticity should not be limited to vertical and horizontal sizing @IaaS level : cloud services should adapt their BL to the context of use
  → PaaS provider duty to offer configuration and runtime environment.

Agenda

Illustrative scenario

- Alice wants to provide a multimedia streaming service.
- She needs the optimal trade-off that avoids over/under fitting of the delivered multimedia.

Illustrative scenario

- Alice wants to provide a multimedia streaming service.
- She needs the optimal trade-off that avoids over/under fitting of the delivered multimedia.
- Alice's idea is to create an "encoding-on-demand" scheme at runtime : too many $(device, context)$ situations to be able to encode multimedia *a priori*.

Illustrative scenario

- Alice wants to provide a multimedia streaming service.
- She needs the optimal trade-off that avoids over/under fitting of the delivered multimedia.
- Alice's idea is to create an "encoding-on-demand" scheme at runtime : too many $(device, context)$ situations to be able to encode multimedia *a priori*.

Overall benefits of taking the end-user context are interesting for both :

1. Alice,
2. Her end-user client,
3. Her cloud provider.

Optimize Alice's OPEX

- Writing elasticity rules in cloud computing are *context agnostic*. E.g. : `R1: Add 1 VM for each slice of 10 clients.`
- Alice would pay VMs to encode Full HD multimedia even for smartphones (or she would have to create as many application as (device,context) pairs possible, which is obvisouly not efficient)

Optimize Alice's OPEX

- Writing elasticity rules in cloud computing are *context agnostic*. E.g. : `R1: Add 1 VM for each slice of 10 clients.`
- Alice would pay VMs to encode Full HD multimedia even for smartphones (or she would have to create as many application as (device,context) pairs possible, which is obvisouly not efficient)

- The PaaS provider could provide Alice's DevOp the possibilty to write different elasticity rules using context $\Rightarrow$ Alice would save VMs. E.g. : `R1: Encode in 640p for each smartphone client.` `R2: Add 1 VM for each slice of 50 smartphone clients.`

Enhance the end-user experience

At the same time, end-users :

- would hardly notice a difference in quality,
- have waited less time (given network conditions),
- are merely like to save batteries (given decoding smaller image size).

Tune the cloud provider localisation/sizing strategies

- By taking into account the end-users context, the PaaS could go as far as adapt itself to this context, or let the cloud user (PaaS client) write rules for PaaS instances management.
- Adapt PaaS running services to the context (e.g. tune memory footprint)
- VMs migration, w.r.t. context elements such as :
  - co-localisation of end users (e.g. events) : empirical results show that latency can be cut by a half [3]).
  - end-users preferences : favour green data centres, or national data centres.

Agenda

Agenda

Context

- Need of a standard to harvest context data and represent it as a datasource in the PaaS so that the cloud-user can use this information in their elasticity rules.
- Problems :
  - PaaS provider are to be interoperable. It is difficult to establish new standard in the area.
  - Cannot modelize all possible context information that the cloud user may want/need

1. `http://peergreen.com/`, based in Grenoble

Context

- Need of a standard to harvest context data and represent it as a datasource in the PaaS so that the cloud-user can use this information in their elasticity rules.
- Problems :
  - PaaS provider are to be interoperable. It is difficult to establish new standard in the area.
  - Cannot modelize all possible context information that the cloud user may want/need
- Solution : Use open vocabularies to give sense to context observation using the Semantic Web. Using Linked Data we can also expressed more expressive queries using linked vocabularies.

1. `http://peergreen.com/`, based in Grenoble

Context

- Need of a standard to harvest context data and represent it as a datasource in the PaaS so that the cloud-user can use this information in their elasticity rules.
- Problems :
  - PaaS provider are to be interoperable. It is difficult to establish new standard in the area.
  - Cannot modelize all possible context information that the cloud user may want/need
- Solution : Use open vocabularies to give sense to context observation using the Semantic Web. Using Linked Data we can also expressed more expressive queries using linked vocabularies.
- This issue of the utmost practical interest is being adressed by an engineer in our team, working closely with Peergreen company [1].

1. `http://peergreen.com/`, based in Grenoble

Big data issue

The PaaS will gather several context observations, sampled regularly. The PaaS is also multi-tenants.

Big data issue

The PaaS will gather several context observations, sampled regularly. The PaaS is also multi-tenants.

That makes quite a lot of semantic data. The issue then is the following :

- How to store that many triples at large scale ?
- How to perform parallel inferences ?
- Can this scale at runtime ?

Big data issue

The PaaS will gather several context observations, sampled regularly. The PaaS is also multi-tenants.

That makes quite a lot of semantic data. The issue then is the following :

- How to store that many triples at large scale ?
- How to perform parallel inferences ?
- Can this scale at runtime ?

- Storage is no longer an issue given the number of graph-oriented NoSQL database on the market, and ongoing efforts to close the gap between NoSQL and the Semantic Web [1].
- The two others are.

Agenda

Distributed reasoning

Three approaches :

- Hardware solution to share memory between nodes [4]. Fast, but low cloud compatibility (Alice would need in-house team and hardware).

Distributed reasoning

Three approaches :

- Hardware solution to share memory between nodes [4]. Fast, but low cloud compatibility (Alice would need in-house team and hardware).
- “Mine is bigger than yours” : load all triples in memory on a single node with large resources. This approach is industry-driven, no guarantee that data will fit in-memory (the number of triples is not known in advance).

Distributed reasoning

Three approaches :

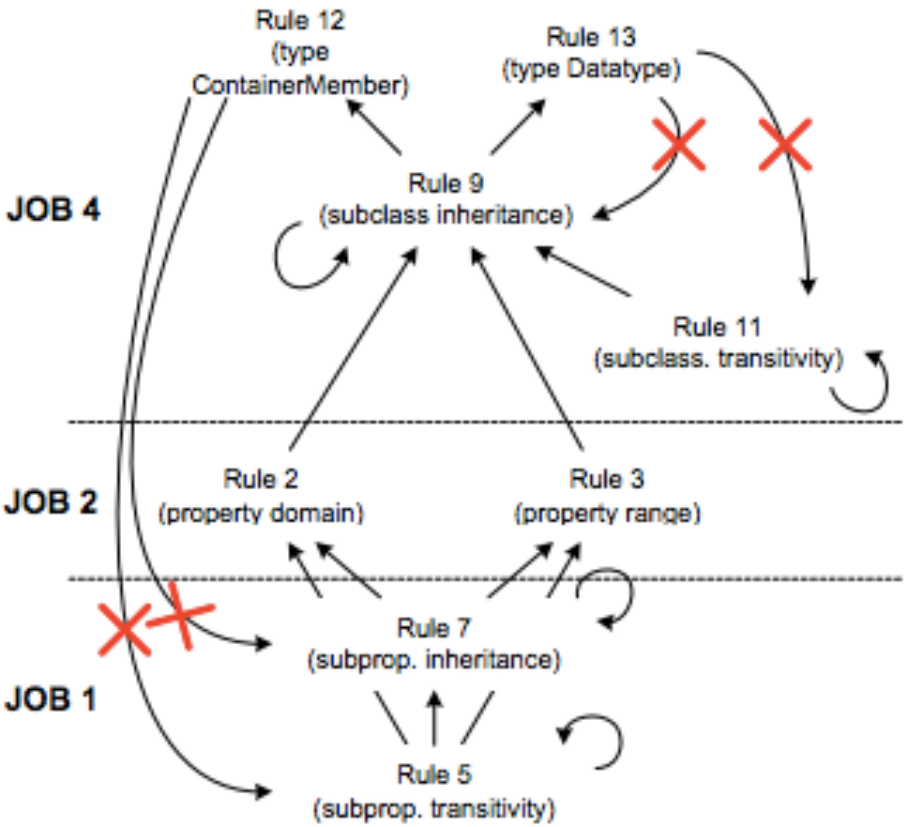
- Hardware solution to share memory between nodes [4]. Fast, but low cloud compatibility (Alice would need in-house team and hardware).
- “Mine is bigger than yours” : load all triples in memory on a single node with large resources. This approach is industry-driven, no guarantee that data will fit in-memory (the number of triples is not known in advance).
- Invent new algorithms to parallelize the reasoning process, i.e. map/reduce jobs that execute a single reasoning rule on all data. E.g. for RDF-S reasoner : WebPie ([6, 7], or MapResolve ([8]).

RDFS

Rule	If...	Then...
1	*s p o* (où *o* est un littéral)	_ :n rdf :type rdfs :Literal
2	*s* rdfs :domain *x* *u s y*	*u* rdf :type *x*
3	*p* rdfs :range *o* *s p v*	*v* rdf :type *o*
4a	*s p o*	*s* rdf :type rdfs :Ressource
4a	*s p o*	*o* rdf :type rdfs :Ressource
5	*p* rdfs :subPropertyOf *p*1 *p*1 rdfs :subPropertyOf *p*2	*p* rdfs :subPropertyOf *p*2
6	*p* rdf :type rdf :Property	*p* rdfs :subPropertyOf *p*
7	*s p o* *p* rdfs :subPropertyOf *p*1	*s p*1 *o*
8	*s* rdf :type rdfs :subClassOf	*s* rdfs :subClassOf rdfs :Ressource
9	*c* rdfs :subClassOf *c*1 *v* rdf :type *c*	*v* rdf :type *c*1
10	*u* rdf :type rdfs :Class	*u* rdfs :subClassOf *u*
11	*c* rdfs :subClassOf *c*1 *c*1 rdfs :subClassOf *c*2	*c* rdfs :subClassOf *c*2
12	*s* rdf :type rdfs :ContainerMembershipProperty	*s* rdfs :subPropertyOf rdfs :member
13	*s* rdf :type rdfs :Datatype	*s* rdfs :subClassOf rdfs :Literal

Map/Reduce for parallel inferencing is tricky

- The dependancy graph of RDFS rules is not acyclic ! Several Hadoop jobs are run multiple times. . . (and it results in duplicates).



"Relation between the various RDFS rules. The red cross indicates the relations that we do not consider" [7].
And it does not scale for more complex fragments.

Hints

However, we can think on some optimizations :

1. Partition the triples (a graph) to minimise map/reduce jobs redundancy (inspired in some way by [9]).

Hints

However, we can think on some optimizations :

1. Partition the triples (a graph) to minimise map/reduce jobs redundancy (inspired in some way by [9]).
2. Streaming new axioms between nodes at each new assertions (avoid batch processing as much as possible).

Hints

However, we can think on some optimizations :

1. Partition the triples (a graph) to minimise map/reduce jobs redundancy (inspired in some way by [9]).
2. Streaming new axioms between nodes at each new assertions (avoid batch processing as much as possible).
3. Rules sequencing depending on axioms used in individuals (rewiring the parallel reasoner depending on the ontology).

Hints

However, we can think on some optimizations :

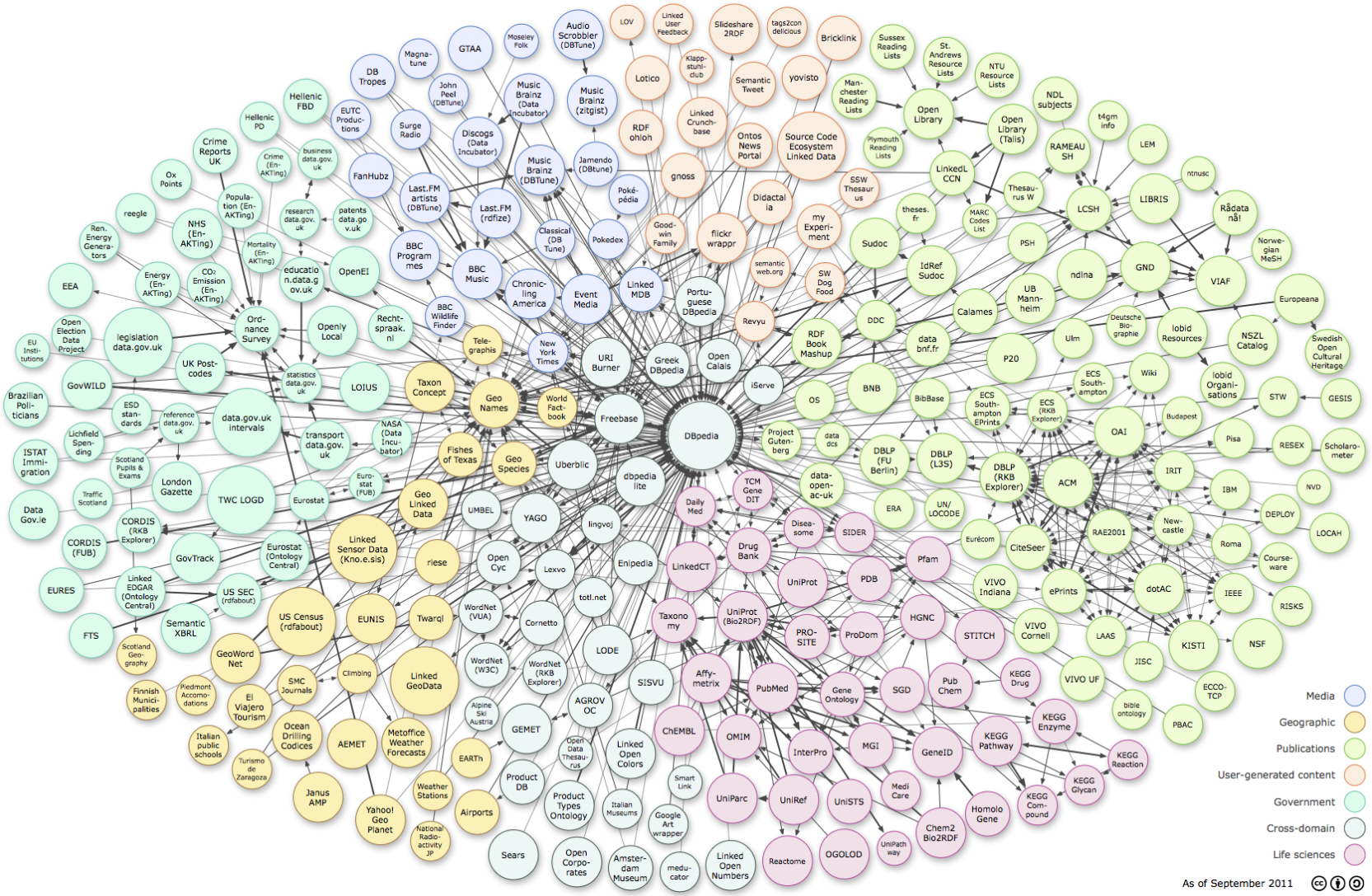
1. Partition the triples (a graph) to minimise map/reduce jobs redundancy (inspired in some way by [9]).
2. Streaming new axioms between nodes at each new assertions (avoid batch processing as much as possible).
3. Rules sequencing depending on axioms used in individuals (rewiring the parallel reasoner depending on the ontology).
4. Distributed in-memory approaches vs batch processing via Hadoop jobs.

Hints

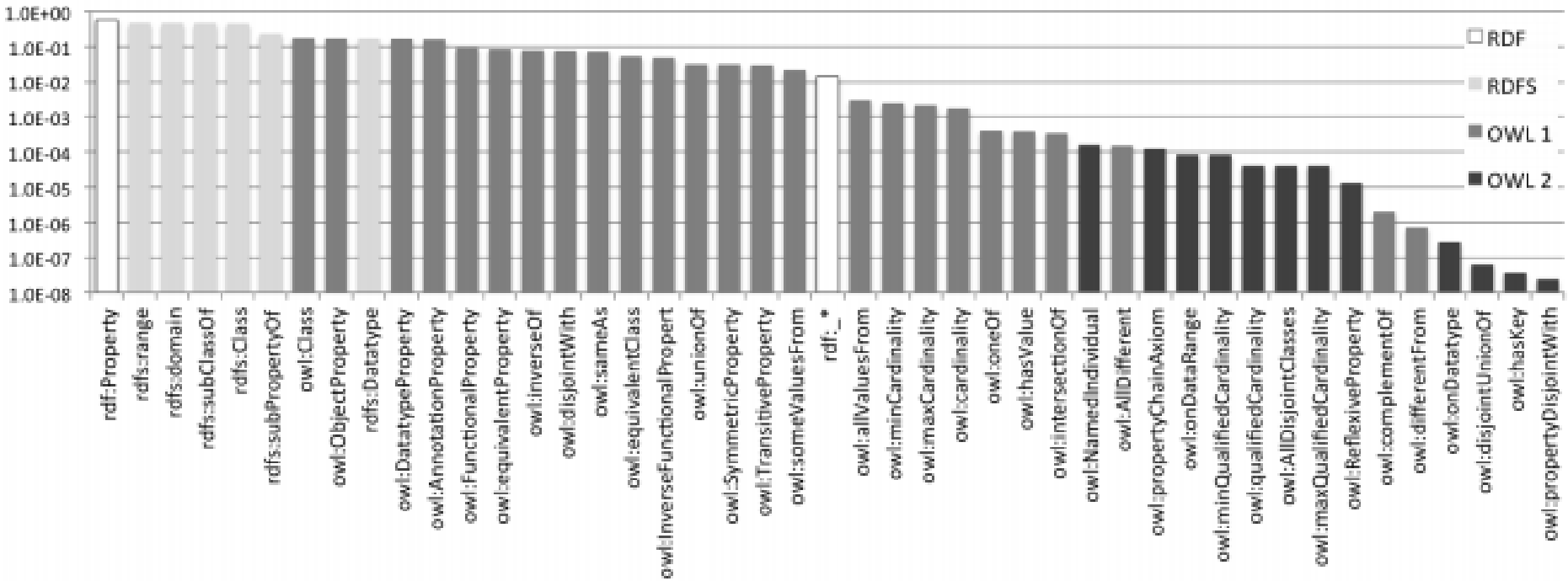
However, we can think on some optimizations :

1. Partition the triples (a graph) to minimise map/reduce jobs redundancy (inspired in some way by [9]).
2. Streaming new axioms between nodes at each new assertions (avoid batch processing as much as possible).
3. Rules sequencing depending on axioms used in individuals (rewiring the parallel reasoner depending on the ontology).
4. Distributed in-memory approaches vs batch processing via Hadoop jobs.
5. Consider other logic fragment, possibly lighter and more adapted.

What logic fragment is useful ?

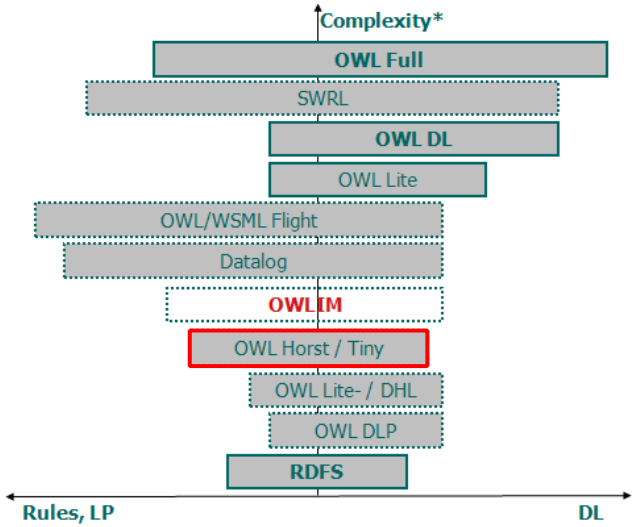


Which logic fragment for the Web of data ?



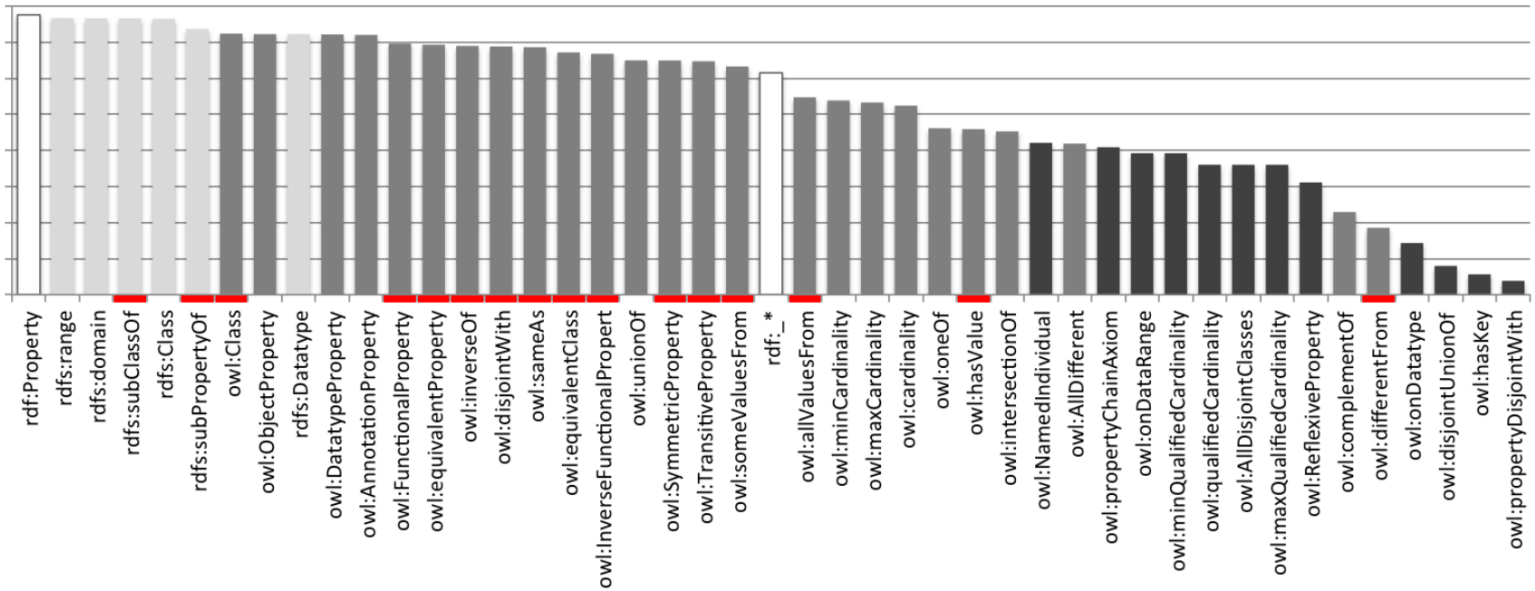
From “OWL : Yet to arrive on the Web of Data ?” [2].

Are we studying the most adapted logic fragment ?



Source : http ://www.ontotext.com/rdfs-rules-owl

- Créé par H. J. ter Horst
- Expressivité entre RDFS et OWL Lite
- Utilisé pour son rapport expressivité/calculabilité



Agenda

Infra in-house

Cloud privé dans les cartons

- Hardware : 128 x (1 vcore + 2 Go RAM)
- OS cloud : Ubuntu/KVM/Openstack
- Configuration :
  - @short-term : Chef-based
  - @long-term : Opencloudware-based :-)
- Utilité : objet d'études (projet opencloudware) et outil pour réalisation (projet Fire).
  - Mais aussi formation et transfert.
- Dispo fin janvier / début février.

Agenda

1. Openclouware R&D project
   - Presentation
   - Use case
   - Technical benefits
2. End-user context
   - Context-awareness
   - Illustrative scenario
3. Big (semantic) data
   - Semantic Web for context-awareness
   - Distributed reasoning
   - Cloud privé
4. Conclusion

Conclusion

From our perspective and participation through the Opencloudware project :

Semantic Web for Cloud Computing

To tackle technical issues : makes it possible to reuse open vocabularies to model context and write business rule for cloud elasticity, without the burden to program it.

Cloud Computing for the Semantic Web

To tackle scientifical issues : intuition is that Cloud Computing can provide scalability to the Semantic Web, however not a solved problem yet.

Questions !

- mail : {christophe.gravier, julien.subercaze, frederique.laforest}@univ-st-etienne.fr
- twitter : @chgravier, @JulienSubercaze, @FredeLaforest

This work has been supported by the French Fonds national pour la Société Numérique (FSN).

References I


Khadilkar, V., Kantarcioglu, M., Thuraisingham, B. M., Castagna, P. “Jena-HBase : A Distributed, Scalable and Efficient RDF Triple Store”. International Semantic Web Conference (Posters & Demos), Boston, MA, USA, 2012


Glimm, B., Hogan, A., Krötzsch, Polleres, A.,“OWL : Yet to arrive on the Web of Data ?”. Linked Data on the Web (LDOW2012), held in connection with 21st International World Wide Web Conference (WWW2012), Lyon, France, 2012


Shakimov, A. , Lim, H., Cox, P. and Cáceres, P. : Vis-à-Vis : Online Social Networking via Virtual Individual Servers. Duke University Technical Report TR-2008-05, October 2008.

References II


Montaner, H., Silla, F., Fröning, F., and Duato, J. “MEMSCALE : in-cluster-memory databases”. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM ’11), pp. 2569–2572, ACM, New York.


Patel-Schneider, P. “Reasoning in RDFS is Inherently Serial, at least in the worst case”, International Semantic Web Conference (Posters & Demos), Boston, MA, USA, 2012.


Urbani, J., Kotoulas, S., Oren, E. & van Harmelen, F., “Scalable Distributed Reasoning using MapReduce”, In Proceedings of the ISWC’09, pp. 634 - 649, 2009.


Urbani, J., Kotoulas, S., Maasen, J., van Harmelen, F. & Bal, H., “WebPIE : A Web-scale Parallel Inference Engine using MapReduce”, Web Semantics : Science, Services and Agents on the World Wide Web, Volume 10, January 2012, pp. 59–75, Elsevier (2012)

References III

Schlicht, A., Stuckenschmidt, H., “Mapresolve”. In Proceedings of the 5th international conference on Web reasoning and rule systems (RR’11), Springer-Verlag, Berlin, Heidelberg,pp. 294-299 (2011).

Ramakrishna S., Prasanna, V.K., “Parallel Inferencing for OWL Knowledge Bases.”, 37th International Conference on Parallel Processing, pp. 75-82 (2008).