Bibliography

[AmazonWS06] Amazon.com. **Amazon Web Services. Infrastructure as a Service**, 2006. http://www.aws.amazon.com/. c

[AmazonWSDoc] Amazon.com. **Amazon Web Services, Documentation**, 2006. http://aws.amazon.com/documentation/. c

[AmazonWSIAM] Amazon.com. **AWS Identity and Access Management (IAM) Documentation**, 2006. http://aws.amazon.com/documentation/. 4.2.4

[Armbrust09] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin and M. Zaharia. **Above the Clouds: A Berkeley View of Cloud Computing**. Technical report, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, 2009. 2.1.1

[Augusto08] C. E. Lara Augusto. **An Infrastructure for Distributed Execution of Software Components**. Master's thesis, *Department of Informatics, PUC-Rio*, 2008. 3.2

[Bruneton06] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma and J.-B. Stefani. **The FRACTAL Component Model and its support in Java**. *Softw. Pract. Exper.*, 36(11-12):1257–1284, september 2006. 2.2.2

[Cerqueira09] C. E. Lara Augusto, L. Marquez, H. Roenick, R. Cerqueira and S. Correa. **SCS - Sistema de Componentes de Software**. *Final Version*, 2009. 3.1

[Chisnall07] D. Chisnall. **The Definitive Guide to the Xen Hypervisor**. Prentice Hall Press, Upper Saddle River, NJ, USA, first edition, 2007. a

[CudennecAnBo08] L. Cudennec, G. Antoniu and L. Boug'e. **CoRDAGe: Towards Transparent Management of Interactions between Applications and Resources**. In *International Workshop on Scalable Tools for High-End Computing STHEC 2008*, pages 13–24, Kos Grèce, 2008. Michael Gerndt and Jesus Labarta and Barton Miller. This work has been

supported by a grant of Sun Microsystems and a grant from the Regional Council of Brittany, France. 2.2.2

[Dean04] J. Dean and S. Ghemawat. **MapReduce: Simplified Data Processing on Large Clusters**. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, *OSDI'04*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. 5.1

[Dearle07] A. Dearle. **Software Deployment, Past, Present and Future**. In *Future of Software Engineering, 2007. FOSE '07*, pages 269–284, may 2007. 1

[DeltaCloud11] D. Apache Software Foundation. **Apache DeltaCloud**, 2011. http://deltacloud.apache.org/. c

[Deng05] G. Deng, J. Balasubramanian, W. Otte, D. C. Schmidt and A. Gokhale. **DAnCE: A QoS-enabled Component Deployment and Configuration Engine**. In *Proceedings of the 3rd Working Conference on Component Deployment*, pages 67–82, 2005. 2.2.2

[Dong10] D. Xu. **Cloud Computing: An emerging technology**. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 1, pages V1–100 –V1–104, june 2010. 2.1.1

[Dubus08] J. Dubus. **Une démarche orientée modèle pour le déploiement de systèmes en environnements ouverts distribués**. These, *Université des Sciences et Technologie de Lille - Lille I*, October 2008. 2.2.2

[Euca2ools08] E. Eucalyptus Team. **Euca2ools, command-line tools compatible with Amazon EC2 and S3**, 2008. http://www.eucalyptus.com/download/euca2ools/. c

[Flissi08] A. Flissi, J. Dubus, N. Dolet and P. Merle. **Deploying on the Grid with DeployWare**. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 177–184, may 2008. 2.2.2

[Fonseca09] E. Fonseca de Andréa. **Monitorando o Ambiente de Execução de Componentes de Software Distribuídos**. Master's thesis, *Department of Informatics, PUC-Rio*, 2009. 5.1

[Grandison10] T. Grandison, E. Maximilien, S. Thorpe and A. Alba. **Towards a Formal Definition of a Computing Cloud**. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 191–192, july 2010. 2.1.1

[Grid5000Fr] INRIA. **Grid'5000**, 2008. 2.2.2

[HangCan10] C. Hang and C. Can. **Research and Application of Distributed OSGi for Cloud Computing**. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–5, dec. 2010. 2.2.3

[Heydarnoori08] A. Heydarnoori. **Deploying Component-Based Applications: Tools and Techniques**. In R. Lee, editor, *Software Engineering Research, Management and Applications*, volume 150 of *Studies in Computational Intelligence*, pages 29–42. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-70561-1_3. 1, 2.2.1

[Hoefer10] C. Hoefer and G. Karagiannis. **Taxonomy of Cloud Computing Services**. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1345–1350, dec. 2010. 2.1.2

[JClouds11] JClouds.org. **JClouds**, 2011. http://www.jclouds.org/. a

[Junior09] A. A. Barbosa Junior. **Deployment of Distributed, Multi-Language and Multi-Platform Component-based Software**. Master's thesis, *Department of Informatics, PUC-Rio*, 2009. 1.3, 3.3, 6

[Kriens11] P. Kriens, R. Nicholson, M. Little, D. Bosschaert and J. S. Rellermeyer. **RFP 133 Cloud Computing**. *Proposed Final Draft*, 2011. 2.2.3

[LacourPePri05] S. Lacour, C. Perez and T. Priol. **Generic Application Description Model: Toward Automatic Deployment of Applications on Computational Grids**. In *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, nov. 2005. 2.2.2

[Libvirt05] L. The Virtualization API. **Libvirt, The Virtualization API**, 2005. http://libvirt.org/. b

[LibCloud10] L. Apache Software Foundation. **Apache LibCloud**, 2010. http://libcloud.apache.org/. b

[Liu10] J. Liu and P. Liu. **Status and key techniques in Cloud Computing**. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 4, pages V4–285 –V4–288, aug. 2010. 2.1.1

[Marriot96] D. Marriott and M. Sloman. **Management Policy Service for Distributed Systems**. In *Services in Distributed and Networked*

Environments, 1996., Proceedings of Third International Workshop on, pages 2–9, jun 1996. 4.2.5, 6

[Merle11] P. Merle, R. Rouvoy and L. Seinturier. **A Reflective Platform for Highly Adaptive Multi-Cloud Systems**. In *Adaptive and Reflective Middleware on Proceedings of the International Workshop*, *ARM '11*, pages 14–21, New York, NY, USA, 2011. ACM. 2.2.3

[Miller01] J. Miller and J. Mukerji. **MDA Guide Version 1.0**. Technical report, Object Management Group, june 2001. 1

[NIST01] P. Mell and T. Grance. **The NIST Definition of Cloud Computing**. *final version*, 2011. 1, 2.1.1, 2.1.2

[NIST02] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger and D. Leaf. **NIST Cloud Computing Reference Architecture**. *final version*, 2011. 2.1.2

[NIST03] M. Hogan, F. Liu, A. Sokol and J. Tong. **NIST Cloud Computing Standards Roadmap**. *final version*, 2011. 2.1.2

[NIST04] L. Badger, T. Grance, R. Patt-Corner and J. Voas. **Draft, Cloud Computing Synopsis and Recommendations**. *draft version*, 2011. 2.1.2

[Nurmi09] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff and D. Zagorodnov. **The Eucalyptus Open-Source Cloud-Computing System**. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124–131, may 2009. a

[OMGDC06] O. M. Group. **Deployment and Configuration of Component-based Distributed Applications Specification, v4.0**. Technical report, Object Management Group, 2006. 1, 2.2.2

[OpenStack11] OpenStack.org. **OpenStack.Open Source software for building private and public clouds**, 2010. http://www.openstack.org/. b

[OpenStackDoc11] OpenStack.org. **OpenStack 2011.2. Cactus, Documentation**, 2010. http://docs.openstack.org/cactus. 4.2.1

[OpenStackIdentity12] OpenStack.org. **OpenStack Identity Service.**, 2010. http://docs.openstack.org/developer/keystone/. 4.2.4

[Paraiso12] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy and L. Seinturier. **A Federated Multi-cloud PaaS Infrastructure**. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 392–399, june 2012. 2.2.3

[Pepple11] K. Pepple. **Deploying OpenStack**. O'Reilly Media, 2011. 4.2.1

[Press01] R. S. Pressman. **Software Engineering: A Practitioner's Approach**. McGraw-Hill Higher Education, 5th edition, 2001. 1

[Prodan09] R. Prodan and S. Ostermann. **A Survey and Taxonomy of Infrastructure as a Service and Web Hosting Cloud Providers**. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 17–25, oct. 2009. 2.1.2

[RimalChLu09] B. Rimal, E. Choi and I. Lumb. **A Taxonomy and Survey of Cloud Computing Systems**. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pages 44–51, aug. 2009. 1, 2.1.2, 2.2

[RimalCh09] B. Rimal and E. Choi. **A Conceptual Approach for Taxonomical Spectrum of Cloud Computing**. In *Ubiquitous Information Technologies Applications, 2009. ICUT '09. Proceedings of the 4th International Conference on*, pages 1–6, dec. 2009. 2.1.2

[Schmid09] H. Schmidt, J.-P. Elsholz, V. Nikolov, F. J. Hauck and R. Kapitza. **OSGi4C: enabling OSGi for the Cloud**. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, *COMSWARE '09*, pages 15:1–15:12, New York, NY, USA, 2009. ACM. 2.2.3

[Seinturier09] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni and J.-B. Stefani. **Reconfigurable SCA Applications with the FraSCAti Platform**. In *Proceedings of the 2009 IEEE International Conference on Services Computing*, *SCC '09*, pages 268–275, Washington, DC, USA, 2009. IEEE Computer Society. 2.2.3

[Strauch11] S. Strauch, O. Kopp, F. Leymann and T. Unger. **A Taxonomy for Cloud Data Hosting Solutions**. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 577 –584, dec. 2011. 2.1.2

[Szy02] C. Szyperski. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002. 1

[Teckelmann11] R. Teckelmann, C. Reich and A. Sulistio. **Mapping of Cloud Standards to the Taxonomy of Interoperability in IaaS**. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 522–526, dec. 2011. 2.1.2

[Voorsluys11] W. Voorsluys, J. Broberg and R. Buyya. *Introduction to Cloud Computing*, pages 1–41. John Wiley and Sons, Inc., 2011. 2.1.1

[Wang08] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer and W. Karl. **Scientific Cloud Computing: Early Definition and Experience**. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 825–830, sept. 2008. 2.1.1

[vonHagen08] W. von Hagen. **Professional Xen Virtualization**. Wrox Press Ltd., Birmingham, UK, 2008. a

A
Appendix

A.1 Cloud Infrastructure Service IDL

The *Cloud Infrastructure Service IDL* is divided in five pages, A.1, A.2, A.3, A.4, and A.5.

Listing A.1: Hashtable Interface

```
1  module CloudInfrastructure {
2
3          struct HashObject{
4                  string name;
5                  string value;
6          };
7
8          struct HashObjectComposed{
9                  string name;
10                 HashObject value;
11         };
12
13         typedef sequence<string> StringSeq;
14         typedef sequence<HashObject> HashObjectSeq;
15         typedef sequence<HashObjectComposed> HashObjectComposedSeq;
16
17
18 };
```

A.2 SCS MapdReduce - Deployment Script

The original deployment script to deploy the WordCount application is displayed into A.6, and A.7.

A.3 SCS MapdReduce on Cloud Infrastructures - Deployment Script

The updated version of the deployment script to deploy the WordCount application on cloud infrastructures is showed in A.8, and A.9.

Listing A.2: Cloud Infrastructure Interfaces(1/4)

```
#ifndef CLOUD_INFRASTRUCTURE_IDL
#define CLOUD_INFRASTRUCTURE_IDL

#include "hashtable.idl"

module CloudInfrastructure{

  /* Instance */
  struct InstanceData{
    string image_id;
    string public_dns_name;
    string private_dns_name;
    string state;
    string key_name;
    string ami_launch_index;
    string product_codes;
    string instance_type;
    string launch_time;
    string placement;
    string kernel;
    string ramdisk;
  };

  struct InstanceInfo{
    string instance_id;
    InstanceData instance_data;
  };
  typedef sequence<InstanceInfo> InstanceInfoSeq;

  /* Image */
  struct ImageData{
    string location;
    string ownerId;
    string state;
    string type_access;
    string architecture;
    string type;
    string ramdisk_id;
    string kernel_id;
  };
  struct ImageInfo{
    string image_id;
    ImageData image_data;
  };
  typedef sequence<ImageInfo> ImageInfoSeq;

  /* AvailabilityZone(cluster)*/
  struct ZoneResource{
    string disk;
    string max;
    string ram;
    string cpu;
    string free;
  };
  struct ZoneData{
    string name_zone;
    ZoneResource zone_resource;
  };
  typedef sequence<ZoneData> ZoneDataSeq;

  struct AvailabilityZoneData{
    string name;
    string state;
    ZoneDataSeq azones;
  };
  typedef sequence<AvailabilityZoneData> AvailabilityZoneDataSeq;

};
#endif
```

Listing A.3: Cloud Infrastructure Interfaces(2/4)

```
#ifndef CLOUD_INFRASTRUCTURE_IDL
#define CLOUD_INFRASTRUCTURE_IDL

#include "hashtable.idl"

module CloudInfrastructure{

  /* Reservation */
  struct ReservationData{
    string owner_id;
    string group_id;
  };
  struct ReservationInfo{
    string reservation_id;
    ReservationData reservation_data;
    InstanceInfoSeq instances;
  };
  typedef sequence<ReservationInfo> ReservationInfoSeq;

  /* Instance */
  interface Instance{

    // Get all instances
    InstanceInfoSeq get_instances();

    // Get a num of instances
    InstanceInfoSeq get_instances_num(
      in unsigned short min_instances,
      in unsigned short max_instances);

    // Get a num of instaces according policies
    InstanceInfoSeq get_instances_policies(
       in unsigned short min_instances,
       in unsigned short max_instances,
       in HashObjectSeq policies);

    // Get a num of instaces from instance_ids
    InstanceInfoSeq get_instances_from_ids(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in StringSeq instance_ids);

    // Get a num of instances according policies from instance_ids
    InstanceInfoSeq get_instances_policies_from_ids(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies,
      in StringSeq instance_ids);

    /* Run Instances */
    ReservationInfo run_instances(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies);

    /* Get the number of Instances */
    unsigned short get_number_instances();

  };
};
```

Listing A.4: Cloud Infrastructure Interfaces(3/4)

```
#ifndef CLOUD_INFRASTRUCTURE_IDL
#define CLOUD_INFRASTRUCTURE_IDL

#include "hashtable.idl"

module CloudInfrastructure{

  /* Instance */
  interface Instance{

    /* Get Instance Ids */
    StringSeq get_instance_ids();

    StringSeq get_instance_ids_num(
      in unsigned short min_instances,
      in unsigned short max_instances);

    StringSeq get_instance_ids_policies(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies);

    StringSeq get_instance_ids_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in InstanceInfoSeq instances);

    StringSeq get_instance_ids_policies_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies,
      in InstanceInfoSeq instances);

    /* Get Public IPs */
    StringSeq get_public_ips();

    StringSeq get_public_ips_num(
      in unsigned short min_instances,
      in unsigned short max_instances);

    StringSeq get_public_ips_policies(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies);

    StringSeq get_public_ips_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in InstanceInfoSeq instances);

    StringSeq get_public_ips_policies_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies,
      in InstanceInfoSeq instances);

  };
};
```

Listing A.5: Cloud Infrastructure Interfaces(4/4)

```
#ifndef CLOUD_INFRASTRUCTURE_IDL
#define CLOUD_INFRASTRUCTURE_IDL

#include "hashtable.idl"

module CloudInfrastructure{

  /* Instance */
  interface Instance{

    /* Get Private IPs - Only for development */
    StringSeq get_private_ips();

    StringSeq get_private_ips_num(
      in unsigned short min_instances,
      in unsigned short max_instances);

    StringSeq get_private_ips_policies(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies);

    StringSeq get_private_ips_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in InstanceInfoSeq instances);

    StringSeq get_private_ips_policies_from_ins(
      in unsigned short min_instances,
      in unsigned short max_instances,
      in HashObjectSeq policies,
      in InstanceInfoSeq instances );

    /* VM operations */
    StringSeq reboot_instances(in StringSeq instance_ids);

    StringSeq terminate_instances(in StringSeq instance_ids);

    boolean show_console(in StringSeq instance_ips);

    /* Get Reservations */

    // Get all reservations
    ReservationInfoSeq get_reservations();

    // Get a reservation from reservation_id
    ReservationInfo get_reservation(in string reservation_id);

  };
};
```

Listing A.6: MapReduce Deployment(1/2)

```
1 -- Deploys a MapReduce component using a deployment plan. Just high level
2 -- operations will be used to ilustrate main features.
3 local oil = require "oil"
4 --oil.verbose:level(1)
5 -- local of the corba idl files installation
6 local SCS_HOME = assert(os.getenv("SCS_HOME"),
7                   "Missing the system variable called SCS_HOME")
8
9 -- client orb
10 orb = oil.init()
11 oil.orb = orb
12 orb:loadidlfile(SCS_HOME.."/idl/deployer.idl")
13
14 orb:loadidlfile(SCS_HOME.."/idl/CosEvent.idl")
15 orb:loadidlfile(SCS_HOME.."/idl/mapReduce.idl")
16 orb:loadidlfile(SCS_HOME.."/idl/schedule.idl")
17
18 oil.main(function()
19   -- obtaining the remote proxies to the DeployManager
20   manager = orb:newproxy("corbaloc::localhost:2500/Deployer_1",
21             "IDL:scs/deployer/Manager:1.0")
22
23   -- ComponentID definitions
24   local masterId = {
25     name = "MasterComponent",
26     major_version = 1, minor_version = 0, patch_version = 0
27   }
28   local reporterId = {
29     name = "ReporterComponent",
30     major_version = 1, minor_version = 0, patch_version = 0
31   }
32   local workerId = {
33     name = "WorkerComponent",
34     major_version = 1, minor_version = 0, patch_version = 0
35   }
36   local schedulerId = {
37     name = "SchedulerComponent",
38     major_version = 1, minor_version = 0, patch_version = 0
39   }
40   local channId = {
41     name = "EventChannel",
42     major_version = 1, minor_version = 0, patch_version = 0
43   }
44
45   -- Deployment plan
46   local plan = manager:create_plan()
47   print("[info] plan created with id='"..plan:get_nickname().."'")
48
49   -- defining a host to place the repository
50   local importer = require "scs.deployer.descriptorhelper"
51   machine0 = importer.search( "hosts", "machine0" )
52   -- creating an ExecutionNode
53   exnode = plan:create_exnode()
54   exnode:set_host( machine0 )
55   -- creating a Container specific for Master
56   masterContainer = plan:create_container("java")
57   masterContainer:set_node( exnode )
58   masterContainer:set_property( "classpath",
59     SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
60     "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )
61
62   reporterContainer = plan:create_container("java")
63   reporterContainer:set_node( exnode )
64   reporterContainer:set_property( "classpath",
65     SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
66     "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )
67
68   channelContainer = plan:create_container("lua")
69   channelContainer:set_node( exnode )
```

Listing A.7: MapReduce Deployment(2/2)

```
-- Defining the components as deployment units
-- EventChannel
channel = plan:create_component()
channel:set_id( channId )
channel:set_container( channelContainer )
-- Reporter
reporter = plan:create_component()
reporter:set_id( reporterId )
reporter:set_container( reporterContainer )
-- Master
master = plan:create_component()
master:set_id( masterId )
master:set_container( masterContainer )
-- Workers
workers,containers = {},{}
for i=1,3 do
  containers[i] = plan:create_container("java")
  containers[i]:set_node( exnode )
  containers[i]:set_property( "classpath",
    SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
    "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )
  workers[i] = plan:create_component()
  workers[i]:set_id( workerId )
  workers[i]:set_container( containers[i] )
  workers[i]:set_args( {SCS_HOME ..
    "/scripts/execute/mapReduce.properties",
    exnode:get_host().ip} )
  workers[i]:add_connection("Channel", channel, "EventChannel")
  workers[i]:add_connection("Reporter", reporter, "Reporter")
  master:add_connection("WorkerServant", workers[i], "WorkerServant")
end

-- Scheduler
scheduler = plan:create_component()
scheduler:set_id( schedulerId )
scheduler:add_connection("ExecutionNode", exnode, "ExecutionNode")
scheduler:set_container( reporterContainer )

-- connections
master:add_connection("Scheduler", scheduler, "RoundRobinServant")
master:add_connection("Channel", channel, "EventChannel")
master:add_connection("Reporter", reporter, "Reporter")

-------------------------------------------------------------------------
-- deployment step, you can call exnode:deploy() altenatively
assert(plan:deploy())
print("[info] plan deployed!")
-- activation step, this call startup in all components
plan:activate()
print("[info] plan activated!")
print("[info] user application actions is starting!")

local masterFacet = master:get_facet("MasterServant")
masterFacet = orb:narrow(masterFacet,
    "IDL:scs/demos/mapreduce/Master:1.0")
masterFacet:submitJob(SCS_HOME.."/scripts/execute/mapReduce.properties")

plan:undeploy()
end)
```

Listing A.8: MapReduce Deployment on the Cloud(1/3)

```
--- Deploys a MapReduce component using a deployment plan. Just high level
-- operations will be used to ilustrate main features.
local oil = require "oil"
--oil.verbose:level(1)
-- local of the corba idl files installation
local SCS_HOME = assert(os.getenv("SCS_HOME"),
                 "Missing the system variable called SCS_HOME")

-- client orb
orb = oil.init()
oil.orb = orb
orb:loadidlfile(SCS_HOME.."/idl/deployer.idl")

orb:loadidlfile(SCS_HOME.."/idl/CosEvent.idl")
orb:loadidlfile(SCS_HOME.."/idl/mapReduce.idl")
orb:loadidlfile(SCS_HOME.."/idl/schedule.idl")

local CloudEngine = require "cloud_engine"
local persist     = require "luarocks.persist"

oil.main(function()
  -- obtaining the remote proxies to the DeployManager
  manager = orb:newproxy("corbaloc::vmDeployManagerIName:2500/Deployer_1",
            "IDL:scs/deployer/Manager:1.0")

  -- ComponentID definitions
  local masterId = {
    name = "MasterComponent",
    major_version = 1, minor_version = 0, patch_version = 0
  }
  local reporterId = {
    name = "ReporterComponent",
    major_version = 1, minor_version = 0, patch_version = 0
  }
  local workerId = {
    name = "WorkerComponent",
    major_version = 1, minor_version = 0, patch_version = 0
  }
  local schedulerId = {
    name = "SchedulerComponent",
    major_version = 1, minor_version = 0, patch_version = 0
  }
  local channId = {
    name = "EventChannel",
    major_version = 1, minor_version = 0, patch_version = 0
  }

  -- Deployment plan
  local plan = manager:create_plan()
  print("[info] plan created with id='"..plan:get_nickname().."'")

  --- Loading policies
  policies = {}
  persist.load_into_table("policies.lua", policies)

  --- Deployment Policies
  local hosting, policy, resource, cloud_deployment_model
  policy           = policies.platform.deployment.policy
  deployment_model = policies.cloud_infrastructure.deployment_model.policy

  -- Instantiating a cloud_engine
  cloud_engine = CloudEngine{}
  -- One Master and Two Workers
  local num_instances = 3
```

Listing A.9: MapReduce Deployment on the Cloud(2/3)

```
if policy == 'getting' then
  -- Getting VM Instances
  vmInstances = cloud_engine.
    instances[ cloud_engine:get_cloud_deployment_model() ].
    cloud_connection:get_instances(num_instances,num_instances,policies)

elseif policy == 'running' then
  -- Starting VM Instances
  vmInstances = cloud_engine.
    instances[ cloud_engine:get_cloud_deployment_model() ].
    cloud_connection:run_instances(num_instances,num_instances,policies)
end

--- Creating vmInstanceDescription to DeployManager service
vmInstancesDescription = {}

--- Getting Instance Description
for k,v in pairs(vmInstances.instances) do
  if type(v)=='table' then
    vm_instance_description = {}
     -- instance_info
    vm_instance_description.instance_info = v
    -- legacy parameters
    vm_instance_description.name = v.instance_id
    vm_instance_description.ip   = v.instance_data.private_dns_name
  end
  table.insert(vmInstancesDescription,vm_instance_description)
end

vmInstances = vmInstancesDescription
exNodes = {}

-- creating an ExecutionNodes
for i=1,num_instances do
  exNodes[i] = plan:create_exnode()
  exNodes[i]:set_instance( vmInstances[i] )
end

-- creating a Container specific for Master
masterContainer = plan:create_container("java")
masterContainer:set_node( exNodes[1] )
masterContainer:set_property( "classpath",
  SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
  "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )

reporterContainer = plan:create_container("java")
reporterContainer:set_node( exNodes[1] )
reporterContainer:set_property( "classpath",
  SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
  "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )

channelContainer = plan:create_container("lua")
channelContainer:set_node( exNodes[1] )

-- Defining the components as deployment units
--- EventChannel
channel = plan:create_component()
channel:set_id( channId )
channel:set_container( channelContainer )
--- Reporter
reporter = plan:create_component()
reporter:set_id( reporterId )
reporter:set_container( reporterContainer )
--- Master
master = plan:create_component()
master:set_id( masterId )
master:set_container( masterContainer )
```

Listing A.10: MapReduce Deployment on the Cloud(3/3)

```
-- Workers
workers,containers = {},{}
for i=2, num_instances do
  containers[i] = plan:create_container("java")
  containers[i]:set_instance( exNodes[i] )
  containers[i]:set_property( "classpath",
    SCS_HOME.."/libs/jacorb/\*:"..SCS_HOME..
    "/libs/\*:"..SCS_HOME.."/libs/luaj/\*" )
  workers[i] = plan:create_component()
  workers[i]:set_id( workerId )
  workers[i]:set_container( containers[i] )
  workers[i]:set_args( {SCS_HOME ..
                        "/scripts/execute/mapReduce.properties",
                        exNodes[i]:get_host().ip} )
  workers[i]:add_connection("Channel", channel, "EventChannel")
  workers[i]:add_connection("Reporter", reporter, "Reporter")
  master:add_connection("WorkerServant", workers[i], "WorkerServant")
end

-- Scheduler
scheduler = plan:create_component()
scheduler:set_id( schedulerId )
for i=1,num_instances do
  scheduler:add_connection("ExecutionNode", exNodes[i], "ExecutionNode")
end
scheduler:set_container( reporterContainer )

-- connections
master:add_connection("Scheduler", scheduler, "RoundRobinServant")
master:add_connection("Channel", channel, "EventChannel")
master:add_connection("Reporter", reporter, "Reporter")

-------------------------------------------------------------------------
-- deployment step, you can call exnode:deploy() altenatively
assert(plan:deploy())
print("[info] plan deployed!")
-- activation step, this call startup in all components
plan:activate()
print("[info] plan activated!")
print("[info] user application actions is starting!")

local masterFacet = master:get_facet("MasterServant")
masterFacet = orb:narrow(masterFacet,
  "IDL:scs/demos/mapreduce/Master:1.0")
masterFacet:submitJob(SCS_HOME.."/scripts/execute/mapReduce.properties")

plan:undeploy()
end)
```