WNoDeS, a tool for integrated Grid and Cloud access and computing farm virtualization

Davide Salomoni, Alessandro Italiano, Elisabetta Ronchieri
INFN CNAF, Viale Berti Pichat 6/2, Bologna, 40127, Italy
E-mail: davide.salomoni@cnaf.infn.it, alessandro.italiano@cnaf.infn.it, elisabetta.ronchieri@cnaf.infn.it

Abstract. INFN CNAF is the National Computing Center, located in Bologna, Italy, of the Italian National Institute for Nuclear Physics (INFN). INFN CNAF, also called the INFN Tier-1, provides computing and storage facilities to the International High-Energy Physics community and to several multi-disciplinary experiments. Currently, the INFN Tier-1 supports more than twenty different collaborations; in this context, optimization of the usage of computing resources is essential. This is one of the main drivers behind the development of a software called WNoDeS (Worker Nodes on Demand Service).

WNoDeS, developed at INFN CNAF and deployed on the INFN Tier-1 production infrastructure, is a solution to virtualize computing resources and to make them available through local, Grid or Cloud interfaces. It is designed to be fully integrated with a Local Resource Management System; it is therefore inherently scalable and permits full integration with existing scheduling, policing, monitoring, accounting and security workflows. WNoDeS dynamically instantiates Virtual Machines (VMs) on-demand, i.e. only when the need arises; these VMs can be tailored and used for purposes like batch job execution, interactive analysis or service instantiation. WNoDeS supports interaction with user requests through traditional batch or Grid jobs and also via the Open Cloud Computing Interface standard, making it possible to allocate compute, storage and network resources on a pay-as-you-go basis. User authentication is supported via several authentication methods, while authorization policies are handled via gLite Argus. WNoDeS is an ambitious solution aimed at virtualizing cluster resources in medium or large scale computing centers, with up to several thousands of Virtual Machines up and running at any given time.

In this paper, we describe the WNoDeS architecture.

1. Introduction

A large computing center requires significant economic investments for the building of its infrastructure, the ongoing acquisition of up-to-date computing, storage and networking resources, the recurring costs needed for its operations and, last but not least, the personnel costs. Such investments should be exploited trying to utilize the available resources as much as possible. Unfortunately, standard computer technologies, software and habits are often not driven by the optimization of resource usage. For example, a critical and sometimes frequent task is the upgrade of the kernel of a system; with modern multi-core, multi-job machines, it is a common procedure to wait until the machine is completely free of jobs before proceeding with the upgrade. Or, when one wants to support a new or customized execution environment, a dedicated subset of the available resources is often devoted for this. Again, when a computing
center needs to support a large user community requiring, for example, specific versions of a
given operating system, or specific software or middleware installed on the computing nodes,
more often than not other communities running at the same computing center either adapt to
this set-up or resort to using dedicated hardware. All these use cases are common in a large,
multi-disciplinary computing center; the overall reason for the waste of computing resources
associated to these cases is that resources are *statically* allocated.

Recent advances in virtualization technologies are rapidly changing the landscape of both local
and distributed computing. Virtualization is greatly contributing in simplifying the management
of computing centers, making it possible to offer services that are easier to use, more dynamic,
and more stable. Virtualization splits real computing resources into the hardware and the
virtualization layers. The former is defined by several types of computing resources like CPU,
storage and network; the latter introduces *dynamism* and *polymorphism* in computing resources.

This paper describes a software called **WNoDeS** [1], an acronym for **W**orker **N**odes **o**n **D**emand **S**ervice. **WNoDeS** is developed by INFN and dynamically allocates virtual
resources out of a common resource pool; it aims to exploit resource usage through sharing
and virtualization, smoothly integrating into existing computing centers. **WNoDeS** bases its
resource scheduling decisions on a Local Resource Management System (LRMS). From the
user point of view, an authentication gateway has been recently introduced in the architecture
in order to grant resource access to users with different authentication systems. **WNoDeS**
allows resource exploitation through local and Grid access interfaces in case batch jobs are to
submitted, through a script-based approach called **VIP** (Virtual Interactive Pools) to support
the local instantiation of customized resources, and through the Open Grid Forum (OGF) Open
Cloud Computing Interface (OCCI) standard [3] to support users requiring generic virtualized
computing resources.

This paper is organized as follows. After an overview of the **WNoDeS** architecture in Section 2,
we describe the access interfaces supported by **WNoDeS** in Section 3. Then, in Section 4, we
detail the typical **WNoDeS** workflow. In Section 5 we focus on the state of **WNoDeS** development
and deployment. Finally, Section 6 presents concluding remarks as well as future directions.

### 2. The **WNoDeS** Architecture

**WNoDeS** has a multi-layer architecture, shown in Figure 1, providing: integrated access to
existing (e.g., local, and Grid) infrastructures; multiple authentication methods; multiple access
interfaces, including Cloud computing; flexible and scalable integrated scheduling policies; on-
demand virtual service provisioning; and access to both local and external resources.

As stated in Section 1, an LRMS represents the key scheduling component of **WNoDeS**, and is responsible for accepting jobs into the computing farm using whatever method is locally
configured and supported. The LRMS is responsible for allocating resources, and since this is
normally a highly tuned, configurable piece of software explicitly designed to efficiently dispatch
jobs, and most likely already in use in a computing center, we decided that the architecture
should alter as little as possible of its configuration. This makes it possible to re-use in most
cases the same LRMS configuration with or without **WNoDeS**, not requiring to change job
submission methods for users, nor to significantly alter logging and monitoring subsystems.
**WNoDeS** currently works with Platform LSF [2], since this is the LRMS that the INFN Tier-1
has been using for the past six years. However, **WNoDeS** is portable to other LRMS; the current
LRMS requirements are the support for pre- and post-execution scripts, and the ability to move
a stopped job from one host to another.

Authentication is internally based on X.509 digital certificates. An authentication gateway
has been developed in order to let users with other authentication methods access resources.
Authentication methods supported by the gateway include X.509, federated services like
Kerberos or Shibboleth or simple username/password mapping. Details on the authentication
gateway are provided in [4]. The authorization layer, defining and enforcing access policies, is based on gLite Argus [5].

While it is not necessary that all systems in a computing farm are managed by WNoDeS, all systems under the control of WNoDeS must have the ability to run VMs. After an extensive set of tests [6], we decided to choose the Linux Kernel-based Virtual Machine (KVM). The KVM hypervisor is therefore installed on every node supporting WNoDeS.

Each compute resource always runs a special VM called bait. The role of the bait is to present the capabilities of a physical system to the LRMS. The bait appears to the LRMS head node as a normal resource, and publishes the maximum number of allowed job slots for the physical system it runs on. This number is dynamic, and typically depends on the characteristics of the VMs. Since the bait is part of the LRMS cluster it will receive jobs, which won’t be executed on the bait, though: the bait will instead immediately stop them and request the underlying hypervisor to create appropriate VMs; it will then dispatch the job for execution onto these VMs once these are ready. If there are no more available resources on the physical machines (for instance, because the physical RAM has been exhausted), the bait will be flagged as closed to the LRMS and will stop receiving jobs. Conversely, the bait will be re-opened to the LRMS when resources to create new VMs become available. VMs are KVM-based and run images defined either by the system administrator, or directly selected by users, if authorized to do so; VMs on a physical node are only created by the KVM hypervisor upon requests from a bait, and can only get jobs by the bait hosted on its own physical machine. Finally, WNoDeS may be configured so that when a job on a VM ends, the VM is either destroyed entirely, or suspended: in the former case it will be re-created from scratch when another job arrives; in the latter it will be resumed when a job of the same type (i.e., with a request for the same execution environment) is received by the bait. The VM images are stored, in read-only mode, on some shared storage, and copied when necessary on to the appropriate hypervisors.

3. Access Interfaces

Users can access resources through local, Grid, Cloud or Virtual Interactive Pools (VIP) interfaces; local and Grid interfaces are used for the handling of batch job submissions; the Cloud interface allows users to instantiate a VM using the OCCI API or through a Web portal; the VIP interface is used by local users to allocate customized resources via simple scripts.

3.1. Local Submission

With local submission, VMs are used to automatically instantiate virtual compute nodes (also called Virtual Worker Nodes, or VWNs for short) on-demand for the purpose of executing batch jobs. Each instantiated VM is customized so that it matches job requirements. No specific action is required in this case by the users; the WNoDeS system is pre-configured with a mapping between user (or group) and VM to be utilized for the VWNs.

Figure 1. The WNoDeS architecture.
3.2. Grid Integration

WNoDeS allows Grid users to select, at job submission time, the virtual image to be loaded on the VM that will execute the job; this method is fully integrated with the Grid gLite [10] middleware. In the gLite Workload Management System (WMS) [7], users can describe the characteristics of their jobs using a well-known formalism called Job Description Language (JDL) [8], where one can specify the requirements, constraints and preferences of a job. The attributes used for describing high-end resources come from the GLUE Schema [9].

The currently deployed GLUE Schema v1.3 foresees the SubCluster entity to provide details of the machines offering execution environments to jobs. This entity refers to a homogeneous set of hosts with respect to a given set of attributes. None of the attributes in SubCluster is suitable for describing something like the virtual image that a host is running. At the time of the definition of the GLUE Schema v1.3, virtualization was not widespread enough to raise use cases for the description of virtual resources in terms of images. While waiting for an evolution of the GLUE schema that will take into account the recent advances in virtualization, we pragmatically decided to make the SubCluster entity fit to our goal. The attribute we have chosen is ApplicationSoftwareRuntimeEnvironment, originally meant to describe a list of software packages installed on the hosts. While this choice could be considered incorrect semantically, we believe that this attribute is, among what we could find in GLUE Schema 1.3, suitable to be used for such a case. The choice potentially leads to name clashes in case users utilize the same attribute for a different reason. To limit such a risk, we adopted a naming convention, pre-pending the string in ApplicationSoftwareRuntimeEnvironment with a known string. This makes it very unlikely that the same name is used with another purpose in mind.

The gLite middleware allows submission to the CREAM [11] Computing Element (CE) through the WMS. With WMS version 3.1, and with jobs submitted using the glite-wms-job-submit command on a User Interface, the CERequirements JDL attribute can be used to specify requirements for CE, as shown in the following example:

```jdl
CERequirements="
  other.GlueCEPolicyMaxCPUTime >= 100 &&
  other.GlueHostMainMemoryRAMSize > 2000"
```

To enable virtual image selection, this attribute must contain the GlueHostSoftwareApplicationRuntimeEnvironment string. As mentioned, WNoDeS adopts a naming convention to understand which of the members of this attribute is requesting a virtual image:

```jdl
CeForwardParameters=
  {"GlueHostSoftwareApplicationRuntimeEnvironment"};
CeRequirements="
  (Member("vm_ethics_slc4",
    other.GlueHostApplicationSoftwareRuntimeEnvironment))
```

WMS forwards user requirements to CREAM, which then interacts with the underlying LRMS through a uniform interface called BLAH. BLAH manages the CERequirements expression and sets environment variables that are eventually made available to the LRMS. WNoDeS eventually uses these variables to properly instantiate the requested virtual image.

3.3. Cloud Integration

WNoDeS provides a Cloud computing interface implementing a subset of the OCCI API [3]. The OCCI API is being developed within OGF to access resources Infrastructure as a Service (IaaS), allowing users to access and manage computing, storage, and network resources using a Uniform Resource Identifier. The API is accessible either programmatically for maximum
flexibility, or via a Web-based application for maximum ease of use. Currently, upon a VM deployment request a "dummy job" is sent to the LRMS, and from there to a bait; the dummy job eventually runs on the allocated Cloud VM. This has the key disadvantage that the Cloud VM is part of the LRMS cluster. The next version of WNoDeS will instead support Cloud VM instantiations so that a Cloud VM is totally oblivious of the LRMS. In case of Cloud allocations, only wall clock time is considered for accounting purposes.

3.4. Virtual Interactive Pool
The Virtual Interactive Pool (VIP) interface is targeted at local users who can, through provided scripts, autonomously create customized computing resources. VIP can be useful for example for the creation of individual user interfaces for interactive analysis or for software development. The provisioned virtual resources match the user requirements in terms of RAM, CPU, network bandwidth and shared filesystems to be mounted. A VIP request goes through the authentication and authorization layers of WNoDeS; a fake batch job, invisible to the user, containing the specified resource requirements is submitted to the local batch system and dispatched to a bait host. Like in other cases, on the bait the batch job execution is immediately stopped and the process to instantiate the VM starts. When the VM is ready, WNoDeS customizes the machine, e.g. creating the requested mount points. The VM can now be used and is therefore made available to the requester. The associated batch job on the bait is resumed and used for accounting purposes; like in Cloud interfaces, only wall clock time is considered for charging resource usage. At logout, WNoDeS automatically kills the fake batch job and destroys the VIP VM. Further details on VIP are provided in [12].

4. Workflow
In WNoDeS, a job (real or fake) is always sent by the LRMS to a bait via the various access interfaces, for the purpose of allocating a resource. The generic workflow for the creation and use of a VM is described below. Handling of Cloud VMs is slightly different, since Cloud VMs won’t be seen by the LRMS, as described in Section 3.3.

(i) The LRMS dispatches the job for execution to a bait VM.
(ii) The bait receives the job. A job starter script stops it, gets the jobs requirements and sends a notification message to a process called wnodes_bait running on the bait, signaling there is a new job to handle.
(iii) wnodes_bait asks a process called wnodes_hypervisor running on the hypervisor (on the same physical node) to provide a VM.
(iv) wnodes_hypervisor creates a new VM (or resumes an existing one) and notifies wnodes_bait when the VM is ready.
(v) wnodes_bait migrates the job to the VM provided by wnodes_hypervisor.
(vi) The job goes through a second job starter script on the VM, which notifies wnodes_bait that the job has started. The job then runs normally on the VM.
(vii) When the job finishes, a post-execution script on the VM notifies wnodes_bait.
(viii) wnodes_bait suspends (or shuts down) the VM.

5. Development and Deployment State
The state of development and deployment for the WNoDeS access interfaces is described in Table 1. WNoDeS has been deployed in production at the INFN Tier1 for the past twelve months, where it currently manages about 2000 VMs, corresponding to ~20% of the available computing power.
Virtualized resources are mainly used for batch jobs coming from WNoDeS local and Grid interfaces, with Cloud and VIP interfaces currently being tested by beta users. There are at the moment about 20 supported VM images; some of them are used to provide support to legacy operating systems (like Scientific Linux 3 or 4), while other have been created upon specific requests of user communities requesting a tailored execution environments. As an example of a use case of direct interest to sites, the INFN Tier-1 was able through WNoDeS to dynamically migrate from Scientific Linux CERN 4 (SLC4) to Scientific Linux 5, without disruption to service, and to provide legacy support for was for SLC4 to collaborations requesting it. Beyond the INFN Tier-1, WNoDeS is currently being deployed at several other INFN sites.

6. Conclusions
WNoDeS is a flexible and scalable system, allowing access to computing center resources via local, Grid or Cloud interfaces. It is capable of sharing computing resources optimally while giving users the possibility to specify different requirements as their execution environments; it also gives the possibility to instantiate resources as services through VIP or through its web-based Cloud application. WNoDeS re-uses several proven technologies, such as Linux KVM, LRMS-based scheduling and glite ARGUS. Upon the basis of the INFN Tier-1 experience, we believe that through WNoDeS sites may optimize usage of their computing resources and offer new, added-value services to their customers.

Specifically, the optimizations in resource usage that, thanks to WNoDeS, have already been observed at the INFN Tier-1 mainly relate to the possibility offered by its flexible and transparent virtualization mechanisms to avoid static partitioning of resources to implement either temporary or permanent set-ups.

For the former case (temporary set-ups), for instance, we prevented reductions of service and resource availability when migrating our compute farm to a new operating system. Jobs from customers who were able or willing to use a new operating system were directed to VMs booted with that system, while jobs from customers who needed or wanted to retain an old operating system were directed to other, "legacy" VMs. However, potentially all resources were available to both categories of customers, through the integration of WNoDeS with the well-known scheduling mechanism of fair-sharing available on all modern batch systems and adopted at the INFN Tier-1 since several years. Fair-sharing of resources within a single unpartitioned farm therefore helped in reducing the number of unused resources and in maintaining a high overall resource utilization. In a traditional computing center supporting tens of different collaborations, on the other hand, this migration would have implied a lengthy period of time when part of the compute farm was running the new operating system, and another part the old one. During this whole period, estimated in about seven months in our case, neither category of customers would have been able to fully exploit all resources, since these would have been statically dedicated to a specific purpose (i.e., running either the new or the old operating system). While for example most of the big customers at the INFN Tier-1, accounting for about 70% of the pledged resources, were able to migrate to the new operating system in about three months, the customers accounting for the remaining 30% of pledged resources took another four months to complete their migration. In a traditional resource center (i.e., without WNoDeS), the potential number of total resources available to either of these groups of customers would have therefore been substantially lower during this seven months period where migration activities

| Interface Type | Development | Deployment |
|----------------|-------------|------------|
| Local          | Stable      | Production |
| Grid           | Stable      | Production |
| Cloud          | Stable      | Testing    |
| VIP            | In Progress | Prototype  |

Table 1. WNoDeS Development and Deployment State.
took place.

For the latter case (permanent set-ups), we were able, for instance, to support an experiment that had custom requirements for its execution environment. In order to perform full detector simulations, the experiment software required running a dedicated database server and hosting read-only condition data on each compute node that the experiment could use. In a traditional computing center, this would have required either installing this database service on all available compute nodes (potentially leading for example to software conflicts and generally increasing the number of services run on generic compute nodes), or dedicating part of the compute nodes to that experiment. With WNoDeS, we simply created a custom VM and configured the system so that such a VM was transparently used by jobs submitted by the experiment. Similarly to the previous case, avoiding static partitioning of the INFN Tier-1 compute farm helped in achieving higher utilization, compared to a traditional set-up, of the available resources for all customers of our center through the integration of WNoDeS with the batch system scheduling mechanisms.

The level of optimization in resource usage that can be achieved depends on job submission patterns. In general, if the pattern is such that for a certain period of time there is no resource contention (in other words, if there are idle resources during that period) and if a single cluster is configured so that all customers can potentially access all resources when there is no resource contention, avoiding static partitioning of an amount X of resources may lead to those X resources being available to all customers during that time period (prioritization could be defined by scheduling algorithms like fair-sharing). Conversely, with traditional, static partitioning those X resources would be dedicated to specific purposes and thus unavailable to other users even when idle, thereby decreasing potential resource exploitation. Finally, it is important to note that this mechanism of optimal resource utilization is valid in WNoDeS not only in the case of batch or Grid jobs but for all the WNoDeS supported interfaces, therefore including Cloud and VIP; this is because, regardless of the external interface presented to users (which may or may not involve "jobs" from the user’s point of view), WNoDeS internally manages resource allocations through jobs handled by a batch system.

These optimizations are fully exploited when all the nodes of a compute farm are managed by a system providing virtualized environments in a way functionally similar to the one provided by WNoDeS. In the current set-up, WNoDeS manages about 20% of all the INFN Tier-1 compute resources. We therefore plan to progressively migrate all of our resources to a virtualized environment. However, already in mixed set-ups where part of the nodes are virtual and handled by WNoDeS and part are real compute nodes, it is possible to offer new, custom services; these new services will naturally only be able to run on virtual nodes, but at the same time the possibility to have the same execution environment on both real and virtual nodes will be retained, thus avoiding the necessity to partition computing resources for the sake of providing new services.

Future WNoDeS developments will focus on expanding usability of novel services such as instantiation of Cloud-type services; standardization of GLUE VM information; integration of virtualized storage; performance and flexibility improvements at the virtualization and scheduling layers; and consolidation of authentication and authorization mechanisms.

References
[1] WNoDeS Web Site, http://web.infn.it/wnodes
[2] Platform Computing LSF, http://www.platform.com/Products/Platform.LSF.Family/
[3] The OGF Open Cloud Computing Interface, http://www.occi-wg.org/doku.php
[4] D. Salomoni, V. Ciaschini, An Authentication Gateway for Integrated Grid and Cloud Access, International Conference on Computing in High Energy and Nuclear Physics (CHEP’10), 2011
[5] gLite Argus, https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework
[6] A. Chierici, R. Veraldi, A quantitative comparison between Xen and KVM, International Conference on Computing in High Energy and Nuclear Physics (CHEP’09), Journal of Physics: Conference Series, 2009
[7] M.Cecchi et al., The gLite Workload Management System, Advances in Grid and Pervasive Computing, Springer Berlin, Heidelberg, Vol. 5529, Series Lecture Notes in Computer Science, pp. 256-268, 2009
[8] F.Pacini, DataGrid JDL Attributes, http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf, 2003
[9] S.Burke, S.Andreozzi, L.Field, Experiences with the GLUE Information Schema in the LCG/EGEE Production Grid, International Conference on Computing in High Energy and Nuclear Physics (CHEP’07), Journal of Physics: Conference Series, Vol. 119, N. 062019, 2008
[10] gLite - Lightweight Middleware for Grid Computing. http://glite.cern.ch/
[11] C.Aiftimieiei et al., Design and Implementation of the gLite CREAM job management service, Future Generation Computer Systems, Elsevier Vol. 26, N. 4, pp. 654-667, 2010
[12] C.Grandi, A.Italiano, D.Salomoni, A.K.Calabrese Melcarne, Virtual pools for interactive analysis and software development through an integrated Cloud environment, International Conference on Computing in High Energy and Nuclear Physics (CHEP’10), 2011