

Worker Nodes On Demand: running batch jobs in customized virtual environments

Davide Salomoni, Alessandro Italiano, Marco Grossi
INFN-CNAF - Viale Bertini Pichat 6/2 - 40127 Bologna - Italy

The problem

In large computing center farms running multi users/applications batch jobs, providing *ad-hoc* execution environments could force the farm administrators to unpleasant choices, like dedicating a subset of the available resources to each user/application requiring such an environment, be it a given O/S, the presence (or absence) of specific programs or libraries, or else. But this has the important negative side-effect that farm administrators cannot generally optimize resource usage, because potentially some resources may remain idle, and a lost CPU cycle is lost forever.

The nowadays common multi-core CPU technology has changed the way users run batch jobs on individual computing servers (or worker nodes). In farms with any sizable number of users, it is extremely frequent to see batch jobs belonging to different users running on the same machine; this is of course desirable for several reasons, although incorrect or unfair resource allocation or usage is possible. For example, multiple process forking in a batch job can make a given user job to be more CPU aggressive than others; inappropriately demonized processes might waste CPU resources, or in some cases generate security concerns; bugs in a user job code might lead to kernel panic or to memory issues, thus affecting or even killing all the others jobs running on the same machine.

On the other hand, it could be very useful, both from a users and a farm administrator point of view, to provide clearly defined and autonomous execution environments *on-demand*; these environments would need to match the user/application requirements for a given batch job.

Proposed solution

The technology developed in the virtualization area has finally achieved a consolidated level, allowing the use of virtual machines in production environment in a stable and productive manner. This technology offers both a high level of efficiency and performance comparable to the one found in a real machine.

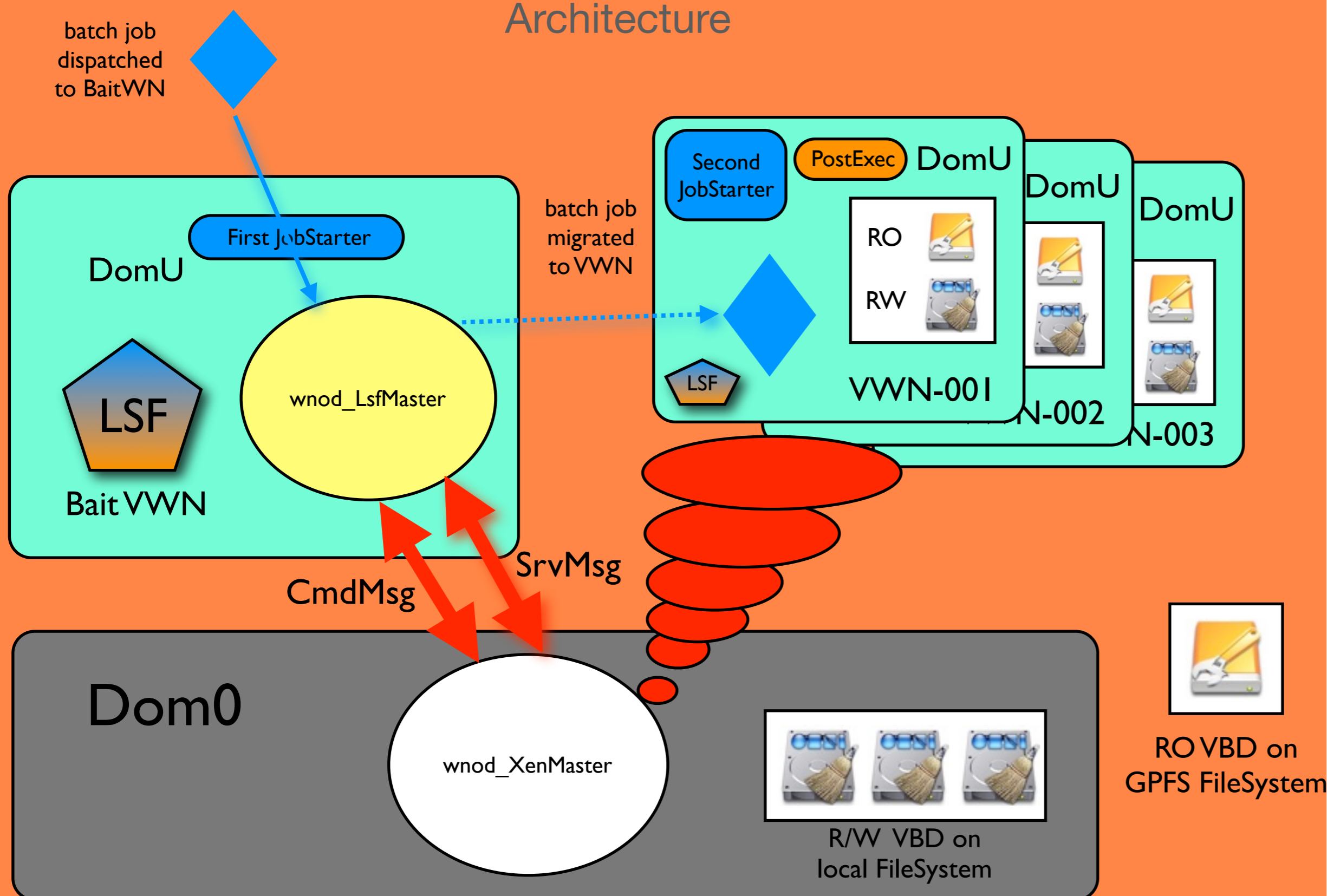
Within this general framework, we have developed an architecture and a software exploiting this architecture, allowing one to run a given batch job on a Virtual Worker Node; the Virtual Worker Node gets created on the spot for the sole purpose of running that batch job, and satisfies the batch job environment requests. This solution tries therefore to address the requirements of both farm administrator and of users/applications. In fact, this allows one to optimize resource usage because it is not necessary to statically dedicate resources anymore. In addition, the solution provides autonomous customized virtual environments, which cannot interact with other batch jobs, thus protecting resources and, in the end, users themselves.

Details of the components

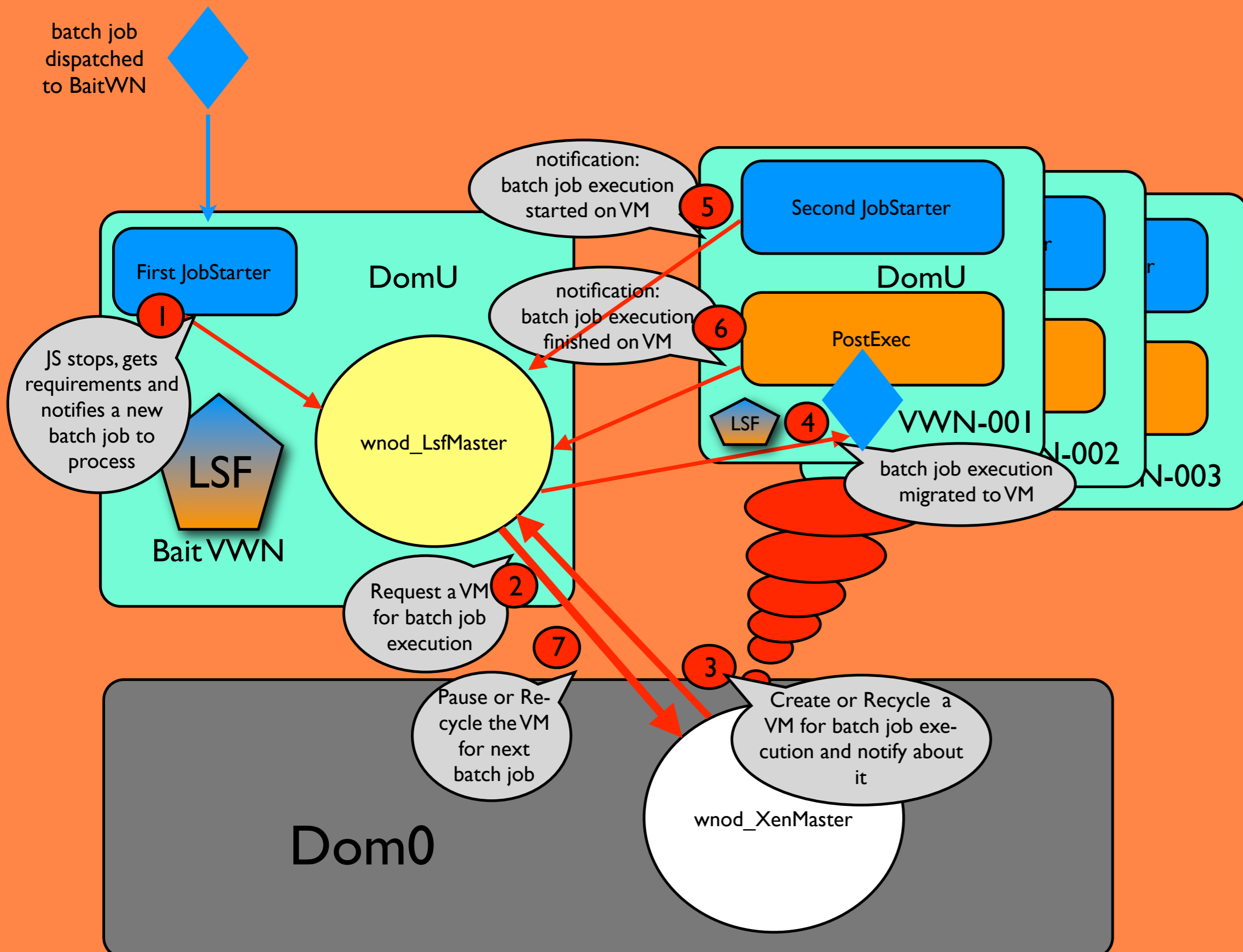
- Batch system:** Platform LSF in our case, although the proposed architecture is not batch system dependent.
- Dom0:** The real machine, running Xen 3.2. It provides the standard environment to run Virtual Machines. It does not belong to the LSF batch system cluster.
- DomU, Bait Virtual Worker Node:** A Virtual Machine which belongs to the LSF batch system cluster, and which acts as a bait for batch jobs. The batch system will dispatch batch jobs to it but they will be executed on Virtual Worker Nodes created on the spot. When there are not anymore resources to create a virtual machine, the bait will be flagged as closed to the batch system; no more jobs will then be dispatched to it. Similarly, the bait will be re-opened to the batch system when resources to create a new virtual machine becomes available.
- DomU, Virtual Worker Node:** A Virtual Machine created on the spot, which will be used to execute the batch job originally dispatched on the Bait Virtual Worker Node. It must match batch job requirements in terms of environment: this may mean operating system, software applications, and hardware resources, like CPU, memory and disk. It belongs to the LSF cluster.
- Job Starter:** A Python script executed when the job starts running. The script must recognize whether it is running on the Bait or on a Virtual Worker Node. On the Bait it gets the job requirements, stops the job and requests wnod_LsfMaster to process it; on the Virtual Worker Node it just notifies that the job has started to run.
- Post Exec:** A Python script which just has to notify wnod_LsfMaster that the batch job has finished to run.
- R/W VBD:** A Virtual Block Device mounted by a Virtual Worker Node and used to provide R/W filesystems (/home, /var/log, /tmp) during job execution. It is stored on the Dom0 machine; there is a R/W VBD for each DomU that could be created on the Dom0.
- RO VBD:** A Virtual Block Device mounted by a Virtual Worker Node which provides a RO filesystem with specific O/S and user application software. It is stored on a shared GPFS filesystem. All DomU of the same type use the same VBD. There is one RO VBD for each possible type of Virtual Worker Node.
- wnod_LsfMaster:** Python Threading TCP Socket server running on a Bait Worker Node which takes care of batch jobs dispatched on the Bait Worker Node.
 - Tasks:
 - Accepts notifications of new batch jobs.
 - Requests a Virtual Worker Node to wnod_XenMaster meeting the job requirements.
 - Communicates only with wnod_XenMaster and with all Virtual Worker Nodes belonging to Dom0.
 - Migrates the batch job on the Virtual Worker Node already created by wnod_XenMaster.
 - Follows the job execution until the job has finished to run.
 - Implements a first security level to avoid any abuse. For example a user cannot destroy a Virtual Worker Node running a batch job belonging to another user.
 - Implements a recovery system in order not to lose any jobs; for example, if some operation fails, the job can be reprocessed or queued.
 - Implements a communication protocol (SrvMsg) to handle exceptions.
- wnod_XenMaster:** Python Threading TCP Socket server running on a Dom0 which takes care of Virtual Machines.
 - Tasks:
 - Handles the Xen operations to create, destroy, pause, un-pause and recycle a Virtual Machine.
 - Communicates only with wnod_LsfMaster, which determines what to do for every batch job.
 - Implements a recovery system in order not to lose any jobs; for example, if some operation fails, it sends a SrvMsg to wnod_LsfMaster to request a reprocessing or re-queuing of the batch job.
 - Implements a communication protocol (SrvMsg) to handle exceptions.

Technical details

Architecture



Process flow for a batch job run on a Virtual Worker Node on demand



Some details about technical issues concerning our proposal

Installation:

In both large and medium-sized computer center farms, an automated tool to handle installation and configuration procedures is normally recommended. In our case we use *Quattor*, a community-supported tool which can easily deploy the proposed solution on our farm with more than 600 servers installed. As usual one can install the Dom0 server with the latest O/S and Xen version without any compatibility issues. Quattor uses local components to configure the system, so with the right components you can create the Bait Virtual Worker Node and the R/W Virtual Block Device and install and configure all the needed software. Naturally one could handle these operations in other ways as well, since the proposed solution is not dependent from the chosen installation system.

Updates:

As described in the box above "Components Details", each DomU of the same type uses the same Virtual Block Device and there is one RO VBD for each type of available Virtual Worker Node. We approach the Updates issue in a simple way. When new updates are ready to be delivered, we create on the spot a *dedicated* Virtual Machine, which mounts the RO VBD in R/W mode: in this way all the installed updates will be available to the next Virtual Worker Node which mounts the updated RO VBD.

Accounting:

Accounting is not really a concern with our proposal, because a Virtual Worker Node is part of the regular farm cluster, so the batch system will continue to account all information about batch jobs as usual.

Monitoring:

We approach the monitoring topic as usual and of course it is not mandatory to monitor all the running DomU, while it is important to monitor the Dom0s. In this new scenario it becomes essential to know which DomUs are running and especially where they are. To address this new issue each operation about a Virtual Worker Node is logged to a database. All stored information is displayed in a smart way on web pages like the one below.

