Linux Clusters Institute: Warewulf Cluster Management Toolkit

Michael Jennings (mej@lbl.gov)
High-Performance Computing Services
Lawrence Berkeley National Laboratory
University of California, Berkeley

A Brief History of the Warewulf Project

• Begun in 2000 by Gregory Kurtzer of LBNL
• “Ware” from “software” plus “wulf” from “Beowulf.”
• Used as part of Scientific Cluster Support pilot program
• Original version used floppy disks
• Pioneered fully stateless node provisioning over network
• Pioneered VNFS format/mechanism
• Provided the underlying code for xCAT 1.x stateless
• On “hiatus” from 2006-2010 during PERCEUS development
• Revived in 2010 with a complete redesign/rewrite
Overview of Previous Versions

• Beowulf Model with unified login/I/O+master node
• Optional 2nd interconnect
• Local/shared storage
• Easily scaled to dozens or hundreds of nodes, possibly more!

Lessons Learned

• Each system was like starting over from scratch
  • Separate node images, software trees, user storage
  • Independent, redundant infrastructure
• Could only leverage economies of scale in one direction
  • Each new cluster was a separate point of administration with its own associated burden (i.e., cost)
  • Only 2 FTE managing a growing number of clusters
• Developing Warewulf while deploying Warewulf had interesting challenges
  • Versions differed between clusters ("version creep")
  • Bugs had to be re-fixed or re-worked-around multiple times
PERCEUS and the Supercluster Model

- Master node for sysadmins only
- Shared login/interactive, storage, and network resources
- Per-cluster storage, interconnect
- IB/RapidBoot provisioning

- Code/UI intermixed
- BDB not ACID
- kexec-based boot
- Can’t mix 32-/64-bit

Introduction to Warewulf 3

- Support for submasters, multiple locations/subnets
- Powerful object-based model with abstract data store & modular plugin framework
Object Model

- Written in 100% object-oriented PERL modules
- Top-level Object with several methods for basic operations
  - Get/set members, property methods, stringify, hashify, clone
- ObjectSet (itself an Object) - container for arbitrary Objects
- Other objects derive from these or children of these
  - DSO (Data Store Object) adds id, timestamp, (un)serialize
    - All objects which must persist derive from DSO
  - Node
  - File
  - Module
  - Event
- Notable exception: DataStore object (singleton)

Abstract Data Store

- Abstract class with defined interface
- Provides methods to persist objects, retrieve objects, and look up objects by specific set of members (called “lookups”)
- Uses Singleton design pattern for efficiency and speed
- Prevents direct access to underlying storage (encapsulation)
- Currently only MySQL is implemented
- Future implementations planned/possible:
  - SQL-based: PostgreSQL, MariaDB
  - NoSQL-based: MongoDB, Cassandra, and/or Voldemort
  - Local: Flat files, BerkeleyDB, SQLite
- Allows Data Store to scale independently from Warewulf itself
Modular Plugin Framework

- Thanks to the power & flexibility of PERL OO, add-on modules can “retrofit” higher-level objects with new capabilities!
  - This means that, e.g., `Warewulf::Node` can “learn” how to be provisioned once `Warewulf::Provision::Node` loads.
  - 3rd-party modules can enhance `Warewulf::Object`, et al.
- The Warewulf Shell (`wwsh`) “learns” new subcommands too!
  - Each user/site need only install the modules they want
  - All non-core modules optional (e.g., `ssh`, `ipmi`, `icr`)
- Event mechanism is also modular – event handlers are dynamic
  - Modules register with Warewulf at load time to handle event(s)
  - Event handlers are called in registration order
  - Handlers may terminate handling of event or may pass it on

Provisioning Concepts

- Bootstrap: Kernel+modules+firmware bundle assigned to nodes
  - Imported into data store from VNFS template via `wwbootstrap`
  - Must contain all kernel files required to boot nodes
- VNFS image: Virtual Node File System; nodes’ root FS image
  - Imported into data store from VNFS template via `wwvnfs`
  - Downloaded via HTTP by `initramfs`; used to populate `tmpfs`
- `initramfs`: Initial root FS image downloaded via PXE/TFTP
  - Combination of a bootstrap object and a shared base image
  - Contains Warewulf `init`, provisioning scripts, and capabilities
- Capabilities: Modular boot-time functionality in `cpio` format
  - Grouped into categories (e.g., `provision`, `setup`, `transport`)
  - `provision-files`, `setupfilesystems`, `transport-http`
Details of Provisioning

1. Node's NIC PXE boots (DHCPs); TFTP's kernel+initramfs
2. Kernel boots and runs WW /\texttt{init} script from initramfs
3. Warewulf initializes network using DHCP or kernel cmdline
4. \texttt{wgetnodeconfig} in default (http) transport queries Node
5. \texttt{provisionhandler} runs series of numbered scripts in initramfs
   a. If \texttt{prescript} property defined on Node, run named File script
   b. Create all partitions and filesystems (default is tmpfs on /)
   c. Download VNFS. Update network config, fstab, runtime.
   d. Copy over /dev and kernel files. Make bootable if needed.
   e. Pull provisioned files and “unmount” filesystems.
   f. If \texttt{postscript} property defined on Node, run it

WWInit

• The \texttt{wwinit} command provides a modular utility to set up and/or verify arbitrary aspects of system configuration
  • Each module is a shell script with a numeric index
  • Each module may have category(es) assigned
  • Each module loads pre-defined shell functions for tests/info
  • Modules can run arbitrary shell commands and act on results

• Users can invoke one particular module, or an entire category, or \texttt{ALL} to run everything (i.e., all available modules)

• The common core provides only a couple simple modules
  • Most \texttt{wwinit} modules are provided by \texttt{warewulf-cluster}
  • Sites may define and contribute their own modules
Intel® Cluster Ready™ & MIC Support

Intel Corporation is the #1 contributor to Warewulf outside LBNL.

• Intel® Cluster Ready™ is a standard/certification program for ISVs/IHVs.
  • ISVs code to the standard and rely on component availability.
  • IHVs and/or system integrators ensure their systems comply.

• The de facto standard open source recipes for the Intel® Cluster Ready™ program are based on Warewulf 3 and were contributed by Intel. Streamlined via the warewulf-icr package.

• Intel® Many Integrated Core (MIC) products offer large numbers of 64-bit Intel Xeon processor cores on a single chip.
  • Object-level support for MICs contributed by Intel.
  • Much of the provisioning and management process is automated via the warewulf-mic package.

Additional Modules

• For a master node, only the common, provision, and vnfs are required. To provision nodes directly (master/submaster) requires common, provision, and provision-server.

• Additional modules may be installed if desired for added functionality:
  • ipmi - ipmi subcommand for wwsh (config and manage)
  • ssh - ssh subcommand for wwsh (parallel command exec)
  • cluster - Contains master node files/utilities for Warewulf clusters, including automatic SSH key generation, user creation, and a series of wwinit modules
  • cluster-node - Provides wwfirstboot, a modular system of scripts executed when a compute node first boots
  • monitor - Work in progress; for now, try warewulf-legacy–*
Future Roadmap

• Monitoring
  • Old Warewulf monitoring code was inflexible and hit scaling limits.
  • Earlier rewrite was not backward compatible and encountered implementation challenges.
  • New effort currently in design/early implementation phase with simple, intuitive plugin interface and an eye toward exascale.

• Power Management
  • Development of advanced algorithms integrating topology/locality awareness with demand/event response

• Scheduling Integration
  • Intelligent manipulation of HPC workloads via unification of environmental inputs to manage Warewulf objects

Warewulf Resources

Warewulf Project
• Web Site: http://warewulf.lbl.gov/
• Documentation: http://warewulf.lbl.gov/trac/wiki/Documentation
• Subversion Repository: http://warewulf.lbl.gov/trac/browser
  https://warewulf.lbl.gov/svn/

Mailing Lists:
• warewulf@lbl.gov
• warewulf-devel@lbl.gov