Distributed Computing with HEP Cloud, GlideinWMS and HTCondor

Marco Mambelli

HEP Cloud developers workshop

March 3 2020
Outline

• Scientific computing
• Jobs and queues
• GlideinWMS
• HTCondor
• Resources
• HEPCloud
• **Scientific computing**
• Jobs and queues
• GlideinWMS
• HTCondor
• Resources
• HEPCloud
HEP experiments require computing!

- Accelerator and detector simulations
- Data reconstruction
- Data analysis

When one computer is not enough

✓ Supercomputer
✓ Cluster (Batch Systems)
✓ Grid
✓ Cloud
Computing resources

- **Supercomputer (HPC High Performance Computing)**
  - Special purpose computer fine tuned to achieve elevated number of operations per second
- **Cluster (batch system)**
  - Collection of parallel or distributed computers which are interconnected among themselves using high-speed networks
  - Local Resource Manager (or batch system) is the software managing the computers in the cluster (e.g. PBS, SLURM, HTCondor, SGE, LSF, ...)
- **Grid (e.g. Open Science Grid)**
  - Combines computers from multiple administrative domains to reach common goals, to solve tasks
  - System that coordinates resources which are not subject to centralized control, using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service
Computing resources (cont)

- **Cloud**
  - Refers to both the applications delivered as services over the Internet and the hardware and system software in the data centers that provide those services
  - aka Elastic computing, available or paid only when used
  - **Software as a Service (SaaS)** is a kind of services where in many users can make use of the software hosted by the service provider and pay only for time its being used (Workday, Slack)
  - **Platform as a Service (PaaS)** provides a high-level integrated environment to design, build, test, deploy and update online custom applications (Amazon orchestration, Google AE)
  - **Infrastructure as a Service (IaaS)** refers to the services provided to the users to use processing power, storage, network and other computing resources, to run any software including operating systems and applications (AWS, Google CE, Fermicloud)
HTC and Problems to solve

• High Throughput Computing
  – use of many computing resources over long periods of time to accomplish a computational task

• Need for more resources
  – Scale to more jobs
  – Access more resources
  – Simplify the management

• Less structured resources and infrastructure
  – Multiple organizations
  – Different systems
  – Less infrastructure
  – Different authentications
• Scientific computing
• Jobs and queues
• GlideinWMS
• HTCondor
• Resources
• HEPCloud
Job, queue
Queues

Efficiency
Queues

Latency vs Efficiency
Queues
Late binding
Pilots
Steady state
Pressure based system

Pilots are expendable
Separation of resource and user problems
Black hole
Overlay system
GlideinWMS

GlideinWMS is a pilot based resource provisioning tool for distributed High Throughput Computing

- Provides reliable and uniform virtual clusters
- Submits Glideins to unreliable heterogeneous resources
- Leverages HTCondor
  - Provides HTCondor pools
  - Uses HTCondor capabilities

Overlay system
Glidein: node testing and customization (used)

- Scouts for resources and validates the Worker node
  - Cores, memory, disk, GPU, …
  - OS, software installed
  - CVMFS
  - VO specific tests
- Customizes the Worker node
  - Environment, GPU libraries, …
  - Starting containers (Singularity, …)
  - VO specific setup
- Provides a reliable and customized execute node to HTCondor
Factory (provider)

- A Glidein Factory knows how to submit to sites
  - Sites are described in a local configuration
  - Only trusted and tested sites are included
- Each site entry in the configuration contains
  - Contact info (hostname, resource type, queue name)
  - Site configuration (startup dir, OS type, …)
  - VOs authorized/supported
  - Other attributes (Site name, core count, max memory, …)
  - Glideins can also auto-detect resources
- Configuration can be auto-generated (e.g. from CRIC), admin curated, stored in VCS (e.g. GitHub)
- Condor does the heavy lifting of submissions.
Frontend (overlap)

- Monitors jobs to see how many Glideins are needed
- Compares what entries (sites) are available
- Requests Glideins from the Factory
- Requests Factory to kill Glideins if there are too many
- Pressure-based system
  - Works keeping a certain number of Glideins running or idle at the sites
  - Glideins requests are gradual to avoid spikes and overloads
- Manages credentials and delegates them to the Factory.
Distributed

- N-to-M relationship
  - Each Frontend can talk to many Factories
  - Each Factory may serve many Frontends
- Multiple User Pools
- High Availability replicas
• Scientific computing
• Jobs and queues
• GlideinWMS
• **HTCondor**
• Resources
• HEPCloud
HTCondor

• HTCondor is a Workload Management System
  – i.e.: batch system or Local Resource Manager

• Open-source batch system implementation
  – Fault tolerant
  – Robust feature set
  – Flexible
  – Local Center for High Throughput Computing (UW Madison)
HTCondor ClassAds

• HTCondor principles: two parts of the equation
  – Jobs: quanta of work
  – Machines: available resources

• ClassAds is a language for objects (jobs and machines) to
  – Express attributes about themselves
  – Express what they require/desire in a match (similar to personal classified ads)
  – Structure
    • Set of attribute name/value pairs
    • Value: Literals (string, bool, int, float or an expression)
**Example Match**

**Pet Ad**
- MyType = “Pet”
- TargetType = “Buyer”
- Requirements =
  - DogLover =?= True
- Rank = 0
- PetType = “Dog”
- Color = “Brown”
- Price = 75
- Breed = “Saint Bernard”
- Size = “Very Large”
- ...

**Buyer Ad**
- MyType = “Buyer”
- TargetType = “Pet”
- Requirements =
  - (PetType == “Dog”) &&
  - (TARGET.Price <= MY.AcctBalance) &&
  - (Size == "Large" | Size == "Very Large")
- Rank = (Breed == "Saint Bernard")
- AcctBalance = 100
- DogLover = True
- ...

Dog == Resource ~= Machine

Buyer ~= Job
HTCondor components

Central Manager

- Negotiator
  - Pull list of idle jobs

- Collector
  - Send Machines properties (classAds)
  - Push keepalives

Submit Node (Job Repo)
- condor_submit <name_of_file>

Execute Node (Machine)
- File Transfer Mechanism
HTCondor components (daemons)
- Scientific computing
- Jobs and queues
- GlideinWMS
- HTCondor
- **Resources**
- HEPCloud
• Scientific computing
• GlideinWMS
• HTCondor
• **Resources**
• Monitoring
• Links
• Demo
Glideins run on Execute Nodes

• This is a machine (worker node, host, node, resource), managed by a (Local) Resource Manager
• More frequently virtual than not
• Characterized by its resources (dimensions):
  – CPUs (or total number of cores)
  – RAM (memory)
  – Disk
• There can be other special resources that the node provides: GPUs, access to devices, software, …
• The Glidein will receive all the node or part of it
• Sometime is not easy to identify everything used by a job
Units of work and resources

Terms used by HTCondor

- **Job**
- **Machine (Startd)**
- **Slot (vm, Starter):** multidimensional partition of a machine
  - Static
  - Partitionable
  - Dynamic

Glidein

- Pilot sent on a Machine (or more)
- Allows partitioning policies
- Job for the Factory
Partitioning in an overlay system

- Dimensions: Cores, Memory, Disk, Lifetime
- The resource (e.g. GPGrid) partitions its Execute nodes
- GlideinWMS further partitions the resources it receives

- E.g. 64 Cores machine split in 16 or 32 cores cluster slots; 16 or 12 cores Glideins in 4 or 2 cores partitionable slots; 2 or 1 core jobs

- Issues
  - Fragmentation (unused)
  - Flexibility (vs Complexity)
  - Under or over provisioning (overbooking or be prudent)
  - Scaling (big slots, fewer slices)
Job and Machine ‘dimensions’

• Job request
  – request_cpus: number of cores, integer, default 1.
  – request_disk: amount of disk space in Kbytes, default to sum of sizes of the job's executable and all input files (or image size)
  – request_memory: amount of memory space in Mbytes, default to executable size

• Machine
  – Cpus: number of cores, integer, by default the available cores
  – Disk: amount of disk space on this machine available for the job in KiB, by default the available space
  – Memory: amount of RAM in MiB in this slot

• Over and Under provision are possible
Units of work and resources
• Scientific computing
• Jobs and queues
• GlideinWMS
• HTCondor
• Resources
• HEPCloud
HEPCloud Facility

• Provision resources
• Facilitate access to resources

• Requirements
  – Multiple resource types
  – Multiple routing options
  – Multiple Job types (single/multi core, GPUs, MPI)
  – Multiple submission models to support more resources
  – Facilitate testing and support
  – Replace the GlideinWMS Frontend
Requirements

• Multiple resource types
  – Grid computing resources
  – Institutional cluster
  – Supercomputers
  – Commercial clouds
  – Private clouds

• Multiple routing options
  – Single point
  – Job directed routing
  – Enable/disable/prioritize resource types
  – Enable/disable individual Supercomputers, Clouds
  – Shape traffic (limits, throttling)
Requirements (cont)

- Multiple Job types
  - Single/multi core
  - GPUs and special resources
  - Singularity and containers
  - MPI

- Multiple submission models to support more resources
  - Pilot based
  - Direct

- Facilitate testing and support

- Replace the GlideinWMS Frontend
  - Out of the box solution
Decision Engine

- Decision Engine
- Decision Engine Framework
- Decision Engine Module
  - Source
  - Transform
  - Logic engine
  - Publisher
- Decision channel
- Data block and Data space
Extra slides
HTCondor building blocks in Glidein WMS

- The Factory works with an HTCondor pool, WMS pool, to submit Glideins to different resources.
- The HTCondor Glideins are pilots that launch a startd that registers on a second HTCondor pool, User pool.
- User jobs are matched and execute on the resources.
- The Frontend monitors the user schedds and notifies the Factory about the need for more Glideins.