CC-NIE Integration: Bringing Distributed High Throughput Computing to the Network with Lark

1. Introduction

High Throughput Computing (HTC) is a computational research technique that focuses on maximizing the science throughput for a set of computing resources, as opposed to getting a task done in the shortest period of time. In the last decade, the focus of high throughput computing has shifted from scavenging resources on the local campus to Distributed High Throughput Computing (DHTC), where the campus researcher can draw on resources from around the nation or globe. The local resources still serve as a critical "home base" for the researcher, but the computational resources available can be greatly increased. We like to think of this as "Submit Locally, Compute Globally." As DHTC often turns a computationally-bound problem to a data-bound problem, network connectivity and sharing from the campus to the national cyber-infrastructure is critical.

Strong examples of DHTC include the Holland Computing Center (HCC) campus grid, Purdue's DiaGrid project [1], and the Open Science Grid (OSG) [2, 3]. One common technological component underlying each example is the Condor software [4, 5]. Condor serves both as a backbone batch scheduler for these infrastructures and as a platform for research into DHTC.

An interesting aspect of Condor is that each pool of nodes does not need to exist independently. A Condor scheduler can send jobs directly to either a local worker node or to a worker node in a remote pool. This technique connects multiple campuses in DiaGrid and the HCC Campus Cluster. The glideinWMS software [6], used by the OSG, has allowed the Compact Muon Solenoid (CMS) [7, 8] experiment to construct a single Condor overlay pool spanning dozens of clusters throughout the globe.

Despite these accomplishments, Condor is a consumer of the network infrastructure, and does little integration and management of the network layer. Condor assists in sharing computing resources, but not network resources. Scheduling decisions in Condor are done without regard to the underlying network capacity. As a result, the OSG Council stated that incorporating knowledge of the network layer into scheduling is a key DHTC research challenge for the next five years [9]. The Worldwide LHC Computing Grid Technical Evaluation Group (WLCG TEG) also recommends that WLCG become more network aware [10].

For example, the Condor scheduler may match jobs with large input files to a remote node with little bandwidth. GlideinWMS makes decisions about the number of CPU resources to request from each site based on the number of jobs in the workflow. No consideration is made in its planning as to whether the network can support the number of jobs.

In the “Any Data, Any Time, Anywhere” (AAA) project [11], jobs assigned to busy OSG processing sites are selectively moved to sites with idle CPUs. We call this *overflow*. Input data is handled by performing remote-I/O from the original site to the destination. Currently, AAA will request overflow slots without regard to the network conditions. Clearly, the incorporation of network scheduling capabilities will allow us to better regulate based on network conditions.

Software-defined networking has become popular for managing large data flows between two sites. Software-defined networking allows upper layers of software to manage networking gear such as routers and switches directly, through standard protocols such as OpenFlow [12]. To date, this network management has not made it to the HTC layer. Most applications involving end-hosts utilize virtual machines, not OS-level processes such as a batch job. Data intensive jobs have proven to be a challenge in terms of the bandwidth they consume, and the complex network topologies they must navigate. Smaller sites tend to put the computing resources behind a campus firewall, which provides insufficient throughput to the wider world. Jobs running at remote sites typically do not have the same level of access as those running inside the trusted network. The lack of publically accessible IPv4 addresses means that jobs need to traverse a NAT bottleneck to access external data servers.

For management capabilities, we demonstrated in 2011 that Condor could interact with the recently added mechanisms in the Linux kernel to provide a per-job network device [13]. We believe this prototype will provide a starting point to have Condor manage the network layer, and keeps this proposal from being primarily about software development. Lark will investigate the creation of a management layer for realizing a job’s stated policy into an actual network topology. Work will be done to turn the prototype code into a production-quality Condor feature, and we will demonstrate the implementation of several job-policy-based network topologies. By enabling network topology individualized at each compute job, finer-grained tradeoffs between performance and security can be realized. For instance, firewalls only need to be bypassed by individual jobs that demand high-bandwidth. We will conduct studies to determine the host-level performance costs of this approach.

In addition to providing a network test bed for HTC networking technologies, Lark will aim to provide network-aware scheduling for the Condor scheduler and the glideinWMS system by integrating perfSONAR data [14]. The initial customer for the glideinWMS improvements will be the AAA project, as a mechanism to regulate the amount of resources requested for each site based on network status.

Both the network-aware scheduling and the Condor-managed networking will serve as plumbing for future investigations along these lines, as they provide Condor with a new “hook” to interact with the networking layer.

2. About the Collaboration

UW and UNL have a history of working together, and working in this area, which suggest that, if funded, this proposal has a high likelihood of success. Both are part of the

Open Science Grid (OSG). UW serves as the lead institution and Bockelman is the Technology Area lead. Both sites are also CMS “Tier-2” computing sites. This grant provides a medium-sized high-throughput computing cluster, requires high-speed connectivity and participation in the perfSONAR network-monitoring infrastructure. The AAA project, one of the initial beneficiaries of Lark, includes both UW and UNL. UNL is the lead institution of AAA.

This small collaboration will be a satellite of the larger OSG collaboration. The OSG represents a significant NSF investment in DHTC. By directly advancing OSG’s research agenda, we believe Lark will have a broad impact on computational science.

Both collaborators have been working with advanced networking. The NSF, through the STCI program, has funded the Condor project to implement IPv6 within the core Condor daemons. This proposal would result in the first wide-scale, multi-site deployment of Condor on IPv6. Routing jobs between sites will depend on the local campus infrastructure’s existing programs to enable IPv6 connectivity. The campuses have a long history in participating in research networking. Both campuses are participants in the DYNES project [15], and UNL was one endpoint in the first public demonstration of the DCN dynamic circuit networking technologies that predated DYNES. As this project will involve end-to-end network circuits between sites, the existing partnership between the Condor Project and HCC with their respective campus networking organizations will be an essential ingredient to Lark. Accordingly, a portion of the UW effort will be fulfilled by a campus network engineer involved with DYNES and intimately familiar with the UW’s backbone configuration and WAN connectivity. At UNL, Attebury has worked closely with the university networking team and is the site contact for DYNES.

Underpinning the technical execution of this project is the Condor technology. The Condor project, with over 20 years of history guided by Miron Livny, is a recognized leader in providing High Throughput Computing technologies. As the most senior staff researcher and Technical Lead for the Condor project, Lark Co-PI Tannenbaum is the right person to lead this effort on the UW side. He is responsible for Condor’s technical architecture and implementation, and has considerable network experience including formerly serving as Technology Editor for the monthly CMP Media publication Network Computing. While UW is the home of the Condor project, Lark PI Bockelman has been a very successful external contributor of new features in the last two years. Interaction with the Condor team will be essential, as we will depend on their existing efforts to finish Condor’s IPv6 support and the implementing of our scheduler improvements in order to minimize any software development within Lark.

3. Summary of Project

We break the project deliverables into three subsections: deploying an advanced networking test bed for HTC, network-aware scheduling, and software-defined networks. The work in each area is outlined below.

Advanced Networking Test-bed for HTC (Lark-ANT)

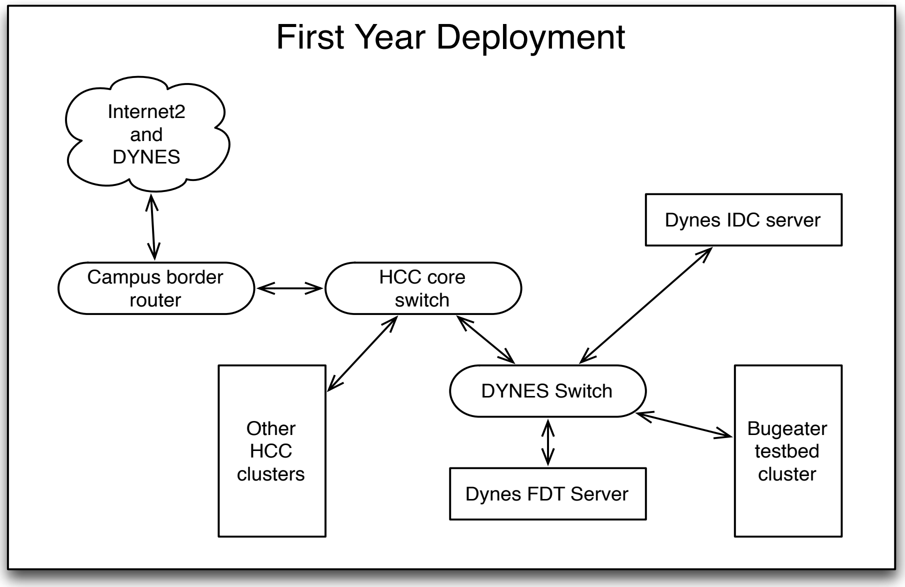
We propose incorporating test clusters at UNL and UW with the DYNES project's current networking test bed into an advanced networking test bed for HTC. The test clusters, from the existing "Build and Test Lab" at UW [16] and "Bugeater" cluster at UNL, will be connected to the DYNES network and be IPv6-enabled. They will be used as a platform for developing the Lark software and as a staging ground before improvements are deployed to the production clusters. The production clusters associated with Lark will be the CMS Tier-2 at UNL ("Red") [17] and the Center of High Throughput Computing (CHTC) pool at UW [18]. Both clusters are more than 3,000 cores and will be able to demonstrate the networking ideas at scale. See Figure 1 for an illustration of the anticipated UNL deployment.

These test beds will initially be connected using IPv6 to each campus's DYNES switch. It will be the first multi-site test of the Condor IPv6 implementation, and will allow us to feed back issues to the Condor team and gain confidence to put IPv6 in production at the Nebraska Tier-2. As Lark progress, the focus of the Lark-ANT pools will be testing the network-aware scheduling and software-defined networks prior to deploying them. In addition, our documented experiences and learned best practices could serve as a case study for how to bring new networking technologies into production DHTC. Moving forward, the HTC community may have to do this more frequently.

Network-aware scheduling (Lark-collect)

At each CMS site and a growing number of OSG sites, including UW and UNL, there is a perfSONAR service that can perform bandwidth and latency tests with other perfSONAR endpoints in the OSG. The results of these tests are stored in an internal database and have well-defined APIs so they can be queried remotely for current and historical test data. We propose to develop Lark-collect to locate the perfSONAR instance associated with a given OSG site, query the current performance data, and feed it into the Condor ecosystem.

The initial use for this data will be to provide information to the Condor scheduler in an "overlay pool." An overlay pool is a Condor pool consisting of worker nodes from many sites. The pool is presented as a homogeneous cluster to the user, but is actually very heterogeneous in terms of the worker nodes. Some worker nodes may be local to the scheduler, while some may be on a different continent. Lark-collect would upload the



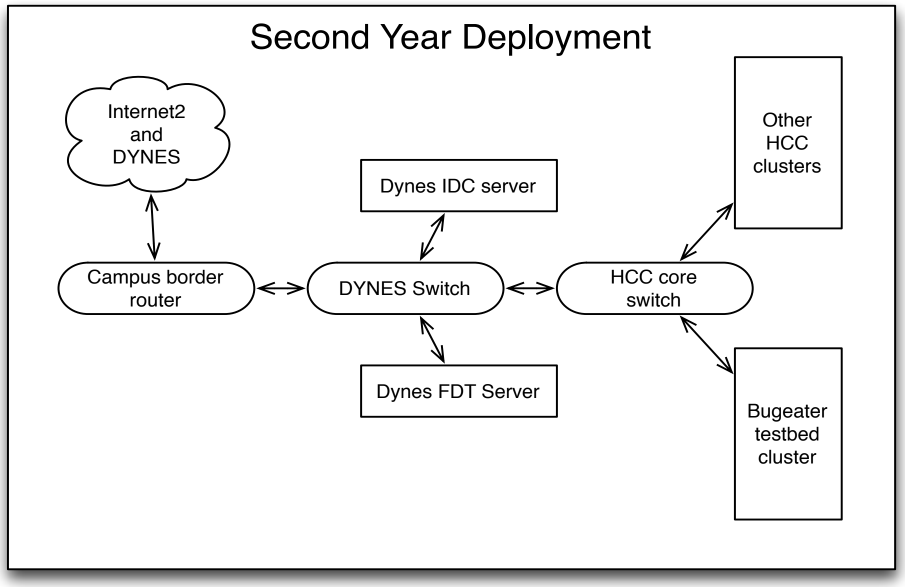


Figure 1. A diagram of the proposed networking configuration at UNL. Technical details have been removed for clarity. During the first year, DYNES and Bugeater will be isolated from the production clusters. In the second year, we will make a transition so DYNES is the primary switch to the campus border router, which will allow deployment of virtual circuits to production.

perfSONAR bandwidth and latency data from the associated site into the Condor collector for each worker node in the system. One potential pitfall in this approach is handling sites with individual resources scattered across a campus. Some nodes may be behind slow NAT boxes while others may be in an unencumbered DMZ, for example. By recording the data at the per-node level, we can improve the data collection in the future. The scheduler, at the time of job submission, can peek at the job's input sandbox (the set of files used as job input) and determine the bandwidth necessary to transfer the sandbox within a given timeframe. Similarly, at the time a job completes, the job description could be updated with the size of the job's output sandbox and a similar calculation could be performed. The scheduler currently limits the number of concurrent transfers to and from worker nodes, giving each transfer the same weight. We propose to expand the scheduler to weigh the active transfers by bandwidth used and to reject matches if it believes it cannot finish the transfer within a given timeframe. See Figure 2. This bandwidth information can also be used to dynamically adjust the number of concurrently allowed sandbox transfers, which is currently expressed to Condor as a static number configured by the administrator.

On the OSG, an overlay pool is constructed by using the glideinWMS software. This software runs a service, the *frontend*, responsible for requesting resources (batch slots, in most cases) at each grid site. The frontend determines the requests based on the number of queued jobs that could match a given site. Accordingly, it currently is able to request many more batch slots than the current network bandwidth could support. Similar to what we propose with the scheduler, we propose to expand the glideinWMS frontend to take into account the network bandwidth available and necessary. This will provide a network-aware regulation mechanism to prevent a user from requesting resources that he will be unable to fully utilize.

The AAA project will assign CMS jobs to an OSG site that has free computational resources, but not the correct CMS input dataset, if the input data can be streamed from elsewhere using remote I/O. This overflow technique is useful to increase CPU utilization in the face of non-optimal distribution of data, at the cost of some per-job efficiency and WAN bandwidth. The AAA infrastructure will statically limit the number of overflow jobs based on conservative estimates. As the limit is input manually, the limits are almost never changed. AAA will also use the network-aware regulation mechanism as a way to automate the overflow limits.

If the interaction with DYNES becomes sufficiently advanced, the next step beyond regulation would be to have Lark reserve and manage bandwidth. If a workflow could be run at a site, given enough WAN bandwidth, Lark could reserve sufficient bandwidth prior to glideinWMS requesting the overflow resources. As the glideinWMS frontend is in charge of planning in the glideinWMS ecosystem, it is the appropriate place to interact with DYNES. To benefit AAA, we would need to know the source site for the remote I/O. This means that DYNES interaction would depend on AAA improving estimates of the likely data source, and therefore we propose to study how to integrate glideinWMS-based regulation mechanisms with the AAA infrastructure towards the end of Year 2.

Condor-Managed Networks (Lark-manage)

In the HTC ecosystem, projects such as the Condor standard universe [4] or Parrot [19] can try to manage a job's I/O or network connectivity. They do this by inserting a shim between the operating system and the running job in order to capture I/O calls and redirect them to a remote host. By using an interposition agent, they can achieve a wide variety of policies, such as allowing transparent file system access to the submit host from the worker node. Interposition agents have their downsides, however, including:

- **Performance**: The interposition is implemented in userspace, not in the kernel, which typically results in an unacceptable performance hit for codes that are not intensely CPU-bound.
- **Reliability**: The interposing agent must emulate the entirety of the complex Linux

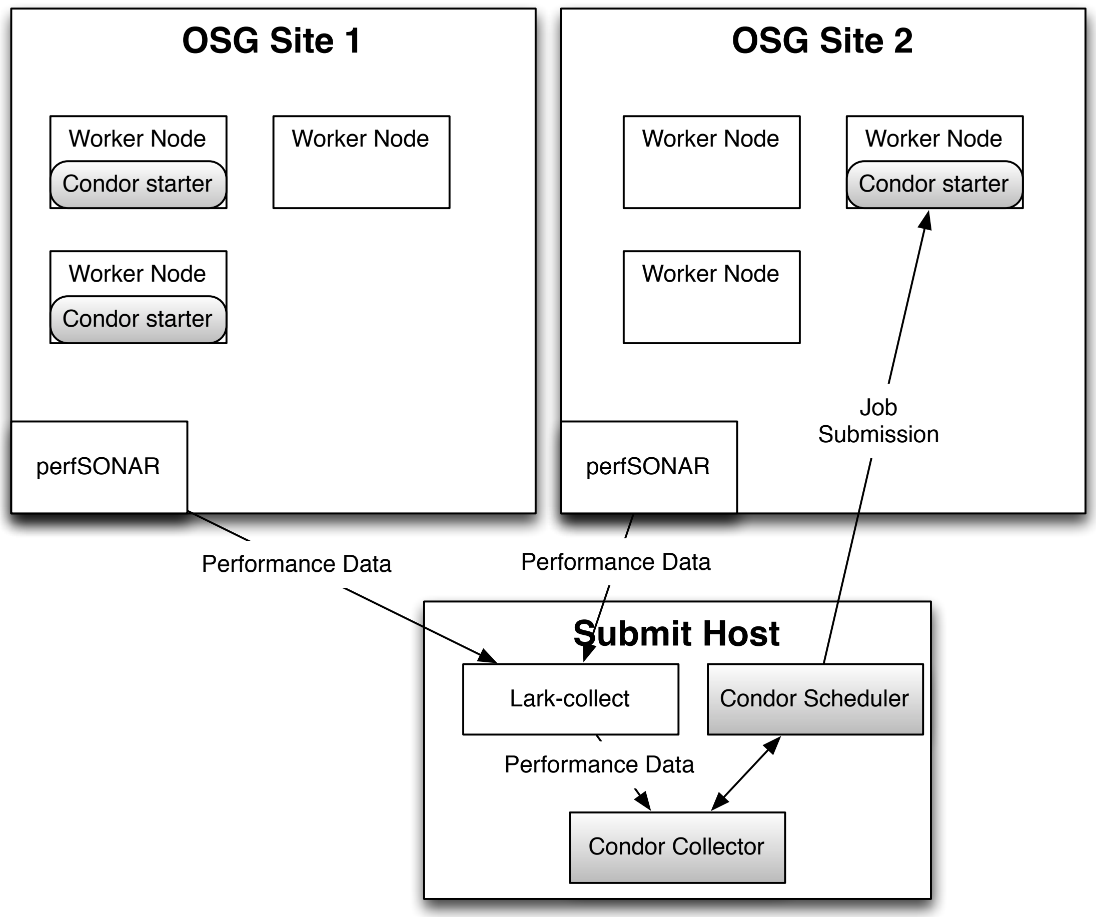


Figure 2. A diagram of how Lark-collect could utilize perfSONAR data to help Condor make scheduling decisions. Here, we diagram an overlay pool (consisting of the Condor components in grey) composed of worker nodes from multiple sites, with differing bandwidth. Lark-collect gathers perfSONAR data from each site and adds the information into the worker node's entry in the collector. When the scheduler matches a job to a host, it can decide whether or not to use that match based on the size of the input files to be transferred from submit host to worker node and the actual worker node.

kernel interface for any generic application, and come up short in many aspects.

- **Security**: The interposing agent often assumes the application is "friendly" and that there would be no adverse impact if the agent makes mistakes or the application bypasses it.

We propose a new approach: combining software-defined networking with the concept of *network namespaces* in Linux. A network namespace defines the network devices with which a set of processes (such as a batch system job, or a virtual machine) is able to interact. Our proposed approach does not require a separate virtual machine with all the image management, performance hits, and updates a virtual machine entails. Even better, it allows us to configure the networking at a *batch job* level as opposed to a network or host level. We have prototyped creating each batch job in a separate namespace, and then adding a *virtual Ethernet device* [20, 21]. The virtual device is a pair of network interfaces that act analogously to a Unix pipe. By making one end the only device accessible to the job's namespace and integrating the other end of the pipe with the host's networking, we can control the job's network access. This prototype work only demonstrated integrating this *mechanism* with Condor, but did not expose any policy. Lark will help guide the mechanism into a Condor software release and, more importantly, write the Lark-manage software to implement the network management *policy*.

Part of the project will be to better determine the policies needed for an organization like the OSG. At minimum we foresee the following:

- *Network accounting*: All of a job's traffic must pass through a single interface. Through the normal Linux routing mechanisms, we can provide for careful accounting of all the network traffic the job generates. We can perform separate accounting for different subsets – such as on campus traffic versus off campus – according to policy.
- *Prioritization and bandwidth limiting*: Through a mechanism similar to network accounting, we can perform QoS and bandwidth limitations at a network-level.
- *Split-network pool*: Based on the job's description – or the policy of the machine's owner – we can segregate a set of jobs (*e.g.*, those involving Health Insurance Portability and Accountability Act (HIPAA) data [22]) from the remainder of the pool by placing the network device on a separate Virtual Local Area Network (VLAN). These jobs would be completely separated from all other local network activity. In addition to protecting trusted jobs with sensitive data, this could prevent untrusted jobs from interacting with core services or having a separate set of policies for a large group of jobs.
- *Network-level flocking*: When a job is sent to a remote pool, we will have the running job rejoin the source pool through joining a dynamically constructed VLAN. This will allow the job to access local cluster resources despite running off-campus. This will be a "capstone" project for Lark, as it will involve end-to-end coordination with DYNES to setup the initial VLAN.

The proposed interaction of the Condor scheduler and Lark-manage is illustrated in Figure 3.

At UNL, the CMS-owned cluster plans to use the split-network pool to limit the aggregate WAN bandwidth used by opportunistic jobs. Without Lark, we can only do this by statically partitioning the cluster, an investment we are unwilling to make for opportunistic jobs. This illustrates how Lark will fundamentally improve our ability to

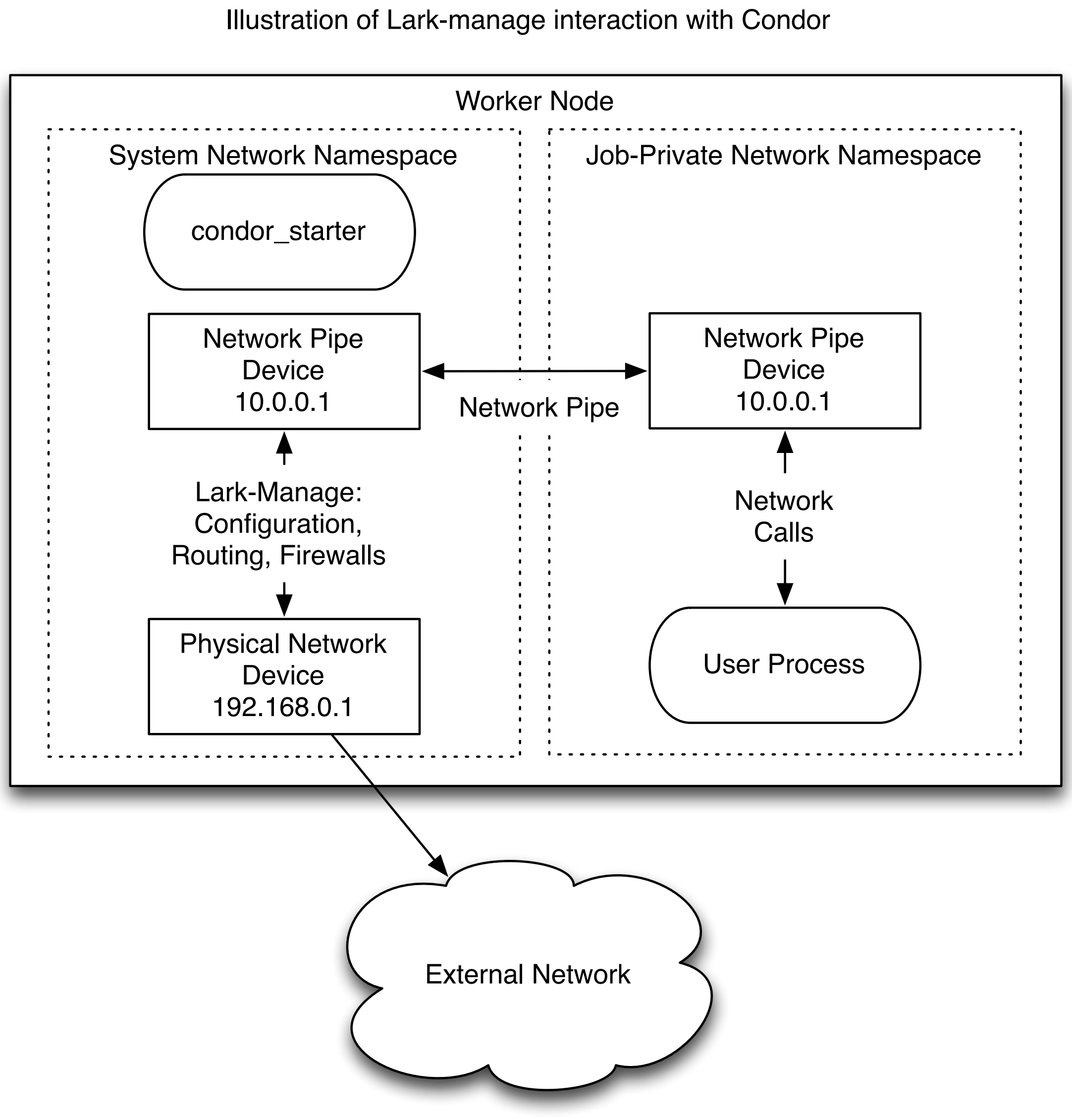


Illustration of Lark-manage interaction with Condor
Worker Node
System Network Namespace
Job-Private Network Namespace
condor_starter
Network Pipe Device 10.0.0.1
Network Pipe
Network Pipe Device 10.0.0.1
Lark-Manage: Configuration, Routing, Firewalls
Network Calls
Physical Network Device 192.168.0.1
User Process
External Network

Figure 3. A diagram of the mechanism used to manage a job's network. We use a Linux feature called "network namespaces" to isolate the job to a single network "pipe device." This device and its pair function in a manner analogous to a Unix pipes, allowing the job to communicate with the outside world in a controlled manner. Lark-manage is responsible for configuring the external pipe device to communicate with the external network. This management can be as simple as configuring a per-job NAT so we can do network usage accounting or performing Layer-2 routing, placing the job on a separate VLAN from the rest of the host.

share resources in HTC.

PIVOT and NEURON Outreach

The PIVOT project at UNL provides small clusters to colleges in Nebraska as a means to make available training in computational research. Having the physical hardware – even though it comes from an otherwise-retired UNL cluster - gives the faculty a sense of ownership and helps motivate the more important training aspects of the program. While working with our initial PIVOT pilot sites, we found that the colleges' staff generally did not have the correct "frame of mind" for supporting research. Their background was in throttling access to student dorms, not enabling sharing between researchers. There was no concept of a "science DMZ" where researchers could collaborate without being hindered by institutional firewalls.

The Nebraska University Regional Optical Network (NEURON) [23] provides an optical network linking universities across the state. Together with PIVOT, it provides an opportunity to deploy Lark technologies and improve research across the state of Nebraska. We plan on connecting PIVOT clusters directly to the UNL data center, implementing a fledgling "science DMZ".

Beyond the science DMZ, we hope this will provide students (ideally undergraduates) and staff the first taste of IPv6 networking and advanced networking by trying to deploy network-level flocking to the PIVOT clusters. We believe that by attempting to push the existing boundaries at these sites, we will help provide the first real-world IPv6 use cases for their IT departments.

4. Project Work Plan

Metrics and Deliverables

We plan the following end-product deliverables for the Lark project. We believe these concrete advances will move the DHTC field forward in incorporating the network layer:

- Condor flocking using IPv6 deployed, tested, and validated between UW and UNL production pools.
- A sustainable ANT pool at each university, connected to the DYNES infrastructure, for use with future networking projects.
- Integration perfSONAR monitoring into the glideinWMS frontend and Condor scheduler layers. Deployment into the AAA and OSG infrastructure.
- A tested and validated hook for Condor to manage the job's network environment, based on the existing prototype software.
- Example network management policies for HTC jobs.
- A report to the OSG on what network management policies are achievable and pragmatic based on the available technologies.
- Deploy a reasonable subset of Lark technologies to the clusters in the PIVOT project.

We propose the following metrics for success, outside the completion of deliverables:

- Number of PIVOT clusters integrated with the UNL data centers.
- Number of software projects integrated with Lark software.
- Number of production jobs run using Lark-manage for the network layer.
- Number of deploys of Lark-collect-enabled glideinWMS submit hosts.

Estimated Schedule

We break down the estimated schedule for the end products of this project below. We approximate by six-month intervals, and have highlighted the items we believe each member of the collaboration will contribute:

Months 1-6:

- **Lark-ANT:** Validate IPv6 routing between UNL and UW. Verify routing can be managed by DYNES. (UNL, UW)
- **Lark-ANT:** Setup ANT pool at UNL. Validate the existing UW NMI pool for ANT. (UNL staff, UW)
- **Lark-collect:** Study how to best collect/query data from perfSONAR repositories. Start design of Lark-collect software. (UNL student)
- **Lark-manage:** Evaluate the Condor network accounting prototype. Determine how to best integrate into Condor release. (UW staff)
- **Lark-manage:** Start design of Lark-manage software. (UNL student)

Months 7-12:

- **Lark-ANT:** Demonstrate IPv6-only functionality for flocking between sites. Feed back to software issues to Condor project. (UW staff, UNL staff).
- **Lark-ANT:** Demonstrate DYNES can construct a usable VLAN between the ANT pools. (UW staff, UNL staff & student)
- **Lark-collect:** Initial prototype of a data aggregator that can feed data into a Condor collector from perfSONAR. (UNL student)
- **Lark-collect:** Work with Condor project to make sure scheduler extensions necessary for estimate bandwidth requirements for stage-in and stage-out exist. (UW staff)
- **Lark-manage:** Implement Lark-manage software. (UNL student)
- **Lark-manage:** Continue integration of Condor network "hooks" into the development series. (UW staff)
- **Outreach:** Extend a VLAN from HCC out to 1 PIVOT cluster. Start planning for IPv6 connectivity. (UNL staff)

Months 13-18:

- **Lark-ANT:** Convert UNL Condor pool to IPv6-only. Deploy mixed-mode IPv4/IPv6 prototype on test bed. (UW staff, UNL staff)
- **Lark-collect:** Have the Condor scheduler take advantage of the bandwidth requirements and availability data. Study what scheduling policies are beneficial (UNL student)

- **Lark-collect:** Improve glideinWMS resource planning mechanisms to utilize the perfSONAR data stored in the Condor collector. (UNL student)
- **Lark-manage:** Implement split-pool use case. (UNL student)
- Additionally, staff at UNL and UW will support UNL students working on the collect and manage projects.

Months 18-24:

- **Lark-ANT:** Deploy mixed-mode support to UNL and UW production clusters. For long-term maintenance, convert the UNL ANT pool to a special subset of the production cluster rather than a standalone cluster. (UW staff, UNL staff)
- **Lark-collect:** Integrate glideinWMS-based regulation mechanism with the AAA infrastructure. (UNL student)
- **Lark-manage:** Implement the network-level flocking policy. (UNL student)
- Staff at UNL and UW will support UNL students working on the collect and manage projects.
- Reports on the usefulness of network-aware scheduling and management to the OSG-ET.
- **Outreach:** Demonstrate network-level flocking to a PIVOT cluster. Assist students working at a PIVOT-participating university to deploy IPv6 connectivity. Extend a VLAN to two PIVOT clusters. (UNL staff)

References

1. DiaGrid, http://www.dia-grid.org/

2. Altunay M, Bockelman B, and Pordes R, "Open Science Grid", available from http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=800

3. Open Science Grid, http://opensciencegrid.org

4. Thain D, Tannenbaum T, and Livny M, "Distributed Computing in Practice: The Condor Experience", *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

5. Litzkow M, Livny M, and Mutka M, "Condor - A Hunter of Idle Workstations", *Proceedings of the 8th International Conference of Distributed Computing Systems,* pages 104-111, June, 1988.

6. Sfiligoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Würthwein F "The Pilot Way to Grid Resources Using glideinWMS", *2009 WRI World Congress on* Computer Science and Information Engineering, vol.2, pp.428-432. doi:10.1109/CSIE.2009.95

7. CMS web site: http://cms.web.cern.ch/

8. The CMS Collaboration, "The CMS experiment at the CERN LHC" . Journal of Instrumentation 3 (08): S08004. doi:10.1088/1748-0221/3/08/S08004

9. OSG Council, "Computer Science Research needed for OSG to be successful in 2020" http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=1106

10. WLCG TEG OPS Face to Face Meeting, Jan 23 2012, Nikhef, Amsterdam, http://indico.cern.ch/conferenceDisplay.py?confId=161833

11. Bloom K, "Any Data, Any time, Anywhere", 2011. Available at http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=1025

12. Open Networking Foundation, https://www.opennetworking.org/

13. Bockelman B, "Improved Site Accounting", 2011. Available at http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=1083

14. PerfSONAR, http://www.perfsonar.net/download.html

15. DYNES, http://www.internet2.edu/ion/dynes.html

16. BaTLab web site: https://www.batlab.org/

17. http://hcc.unl.edu/red/index.php

18. CHTC web site: http://www.chtc.cs.wisc.edu/

19. Thain D, and Livny M, "Parrot: Transparent User-Level Middleware for Data-Intensive Computing", *Scalable Computing: Practice and Experience*, Volume 6, Number 3, Pages 9-18, 2005.

20. http://osgtech.blogspot.com/2011/09/per-batch-job-network-statistics.html

21. https://github.com/bbockelm/condor-network-accounting

22. HIPAA web site: http://www.hhs.gov/ocr/privacy/

23. https://epscor.nebraska.edu/grants/project_nsf_c2.shtml