Lark Annual Report Year 1

(Non-technical information removed)

What are the major goals of the project?

The project-wide goal of Lark is to better integrate the high throughput computing (HTC) layer with the networking layer, primarily using the HTCondor software. Historically, the two layers have existed somewhat independently - while the batch system may implement retries to get around transient network failures, it treated the network as homogenous in availability and speed. This historical configuration is recently starting to break down - HTC resource pools now span site boundaries, meaning the available bandwidth between the submit and worker nodes may vary by an order of magnitude. Further, software defined networking technologies.

Throughout this report, we will focus on the accomplishments and objectives around three goals:

- Allow the HTC layer react to observe and respond to changes in the network layer.
- Allow the HTC layer to proactively change the network configuration through the use of software defined networking (SDN).
- Provide an Advanced Networking Testbed for researchers at University of Wisconsin-Madison and University of Nebraska-Lincoln to further work on SDN and IPv6.

What was accomplished under these goals?

Major Activities

HTCondor Advanced Network Testbed (ANT): Teams at UNL and UW have installed and maintained HTCondor pools on networking testbeds. This has allowed the researchers to validate HTCondor in various IPv6 running conditions and provide feedback for future improvements. ANT has also provided us with the ability to test the integration of HTCondor with OpenFlow-based software defined networking using a variety of networking device and OpenFlow controllers.

Maintain advanced network infrastructure: Lark has served as a coordination point between the two campuses. In particular, networking personnel representing UNL and UW now meet weekly during the Lark meetings to discuss the latest perfSonar results, DYNES progress, and IPv6 work. The network links between UW and UNL now achieve greater than 4Gbps for single-stream TCP.

We have worked to a new piece of software, **lark-collect**, which aggregates monitoring data from the perfSonar ecosystem. This software uses the perfSonar global lookup services to discover the available endpoints in a given group, query their latest point-to-point test data, and upload this data into a HTCondor collector daemon. By providing this data to the HTCondor collector, other HTCondor daemons can start to reason about network performance. We have

begun an activity on integrating this into matchmaking decisions and believe we will have functioning code for perfSonar-influenced matchmaking decisions during Year 2 of the grant.

Lark has provided a meeting place for HTCondor developers interested in better managing I/O and monitoring HTCondor. Our weekly meetings often feature developers not funded by the grant; our team has provided feedback on issues such as managing transfer load, metrics for monitoring I/O, and integrating HTCondor monitoring with Ganglia.

We have made significant progress on **lark-manage**, our software for having HTCondor manage the network layer. We worked on a plugin mechanism for isolating jobs from the host network configuration and allowing a per-job networking. We also have been adding mechanisms for configuring the per-job network device such as DHCP or static addressing. This is the necessary groundwork for allowing the network layer to reason at the granularity of individual jobs, rather than hosts.

The **lark-manage** activity is ahead of the schedule outlined in the grant proposal. The mechanism for integrating jobs onto the network was completed around December 2012, allowing us more time to work on using software-defined-network to make per-job networking decision. This activity centers around on customizing OpenFlow controller configurations to apply HTCondor configuration policy to the network. This activity has utilized a wide variety of switch hardware (from Cisco, Brocade, and Dell) and controllers (Cisco ONE, Floodlight, and POX) to show the approaches are quite flexible.

Specific Objectives

The objectives for Year 1 of this project were explicitly stated within the project proposal. For this year, there were no major additions or deletions from the proposed objectives. Progress was made on all objectives; however, not all were met. The biggest 'misses' in our objectives dealt with interacting with the DYNES project; our plan to catch up is outlined in a separate section.

Advanced Networking

For advanced networking, the biggest objectives dealt with improving IPv6 networking and utilizing the DYNES infrastructure. We were able to validate IPv6 routing between the sites in the collaboration (UW and UNL) and validate the functionality of the IPv6 code by flocking together the two test HTCondor pools. The major impediment to further work is the lack of IPv6-capable DNS at UNL. The rollout of this capability is already a year late; in the meantime, we have been running private, non-authoritative DNS servers. This allows the UW / UNL collaboration to progress on IPv6, but has been a hindrance to our outreach efforts. While work has continued on DYNES throughout Year 1, we were not able to get DYNES-based virtual circuits working. As the first round of DYNES support expired this year, we remain concerned about being able to meet DYNES-related grant objectives. We will continue to reach out to the DYNES project throughout Year 2.

Part of the Lark project supports the UNL's PIVOT outreach project (through a portion of Carl Lundestedt's time). During the grant Year 1, PIVOT provided computing clusters at four

different in-state universities (three of which are primarily undergraduate). At the yearly PIVOT workshop at UNL, there were four faculty, two IT admins, and three students from off-campus during the dedicated cluster-building day and 27 attendees on the general topics day. While we assist the remote sites with networking (assisting with setup, advising deploys for improved connectivity to UNL), we are behind on our goals of having the other clusters connect to us via IPv6 due to the lack of IPv6 DNS at UNL; this has been deferred to Year 2.

Integrating perfSONAR

For Year 1, we set out to design and implement a mechanism to aggregate data from the perfSonar monitoring network into a HTCondor collector. This has been completed. We have also begun to work with HTCondor project to make sure scheduler extensions necessary for estimate bandwidth requirements for stage-in and stage-out exist. These extensions are available as of version 8.1.1 (released in September 2013). These two accomplishments provide central estimates of the data in queue and available bandwidth, the input needed for the Year 2 objectives.

Improving HTCondor

During Year 1, we redesigned the interaction between the network accounting prototype and the HTCondor internals. The prototype now functions as a plugin to the HTCondor condor_starter daemon (which launches the user jobs). Converting this to an optional plugin lowers the bar for adding this to a release and may allow other future groups to contribute their own code. After the evaluation, we also decided to add support for current bandwidth usage estimates; these are updated throughout the job's lifetime.

- Continue integration of Condor network "hooks" into the development series. We have started refining these hooks, but have not committed them into a HTCondor release.

Software Defined Networking

We started the design and implementation of the lark-manage software for integrating HTCondor with software-defined-networking. We wrote a plugin for the POX OpenFlow controller which allows HTCondor to determine network policies for individual jobs. We have shown this controller can work with a variety of different software/hardware (such as OpenVSwitch and switches from Dell). We have begun working on implementing bandwidth management (currently only with OpenVSwitch; to be continued with hardware switches and traffic shaping next year).

Significant results

We believe IPv6 support is a critical feature for HTCondor in the future; for example, CERN is requiring IPv6 support for applications running on its cloud in 2014 as IPv4 address exhaustion is predicted. The Lark project has verified the IPv6 support in HTCondor - even between multiple cluster - and the Lark team has helped guide further IPv6 development in HTCondor.

Our **lark-collect** software can feed query the perfSonar network for monitoring data and feed it into an HTCondor collector. This was not a trivial task - perfSonar's client interfaces are not well documented, nor is there a standard working client. Our student worker spent significant time researching and evaluating client code, as well as feeding back bug reports to the developers.

For our **lark-manage** software, the following is working:

- HTCondor can efficiently create a per-job network device. These network devices can be connected to the outside world via NAT or ethernet-level bridging and can assign themselves IP addresses based on static configuration or DHCP.
- The per-network device can work with an OpenFlow controller module (based on POX) to define per-job network policies. This year, we have implemented three simple network policies with this module - (a) jobs can have no network access, (b) only talk to other jobs, or (c) only talk to other jobs of the same owner and the wide-area network.

We believe this to be a very significant result - each job receives a private network device, allowing network-level decisions to be performed per-job. This further is done without having to use virtual machines - allowing sysadmins to otherwise manage each job as a "normal" batch system job. Network usage accounting has been implemented by simply reading the packet counters of the device.

When the job starts up and the network device is created, HTCondor will perform a network RPC to the OpenFlow module containing the job's ClassAd (the job description). The module then associates the job's device with the description (otherwise unavailable to the network) and apply a centrally-specified policy. For the Cisco Live 2013 demonstration, we did a more primitive variant of this - the wrapper script prior to job startup sent the OpenFlow controller a series of static flow rules to add to the switch based on the job description. While the demonstration was successful, we believe the OpenFlow module approach is superior, as network policy can be administered from a single location.

The per-job network device has allowed us to integrate HTCondor with OpenFlow software defined networking. While we have tested several different OpenFlow controllers, our primary results have been in writing a new HTCondor-specific module for POX. This module receives the job description (a "ClassAd" in HTCondor terminology) and makes network topology and scheduling decisions. We have implemented three policies:

- **No network access**: All packets from the job are dropped.
- **Split-pool**: Each owner's jobs are on a separate network; for example, Jobs 1 and 2 from user Bob can communicate via TCP, but cannot communicate with Job 3 from user Alice.
- **Split-pool with WAN**: Jobs have the same access as in the split-pool case and can additionally route outside the cluster.

For our demonstration at Cisco Live 2013, we showed a variant of the split-pool where jobs had access to an alternate network path and file server based on the job description.

How have the results been disseminated to communities of interest?

Throughout Year 1, we have tried to disseminate the preliminary results of Lark at workshops, meetings, and colloquiums. Preliminary results of the Lark mechanism and job-specific routing

were presented at CiscoWorld 2013 in London by Tannenbaum. The Lark project presented goals and progress to date to the high throughput computing community at HTCondor Week 2013. One of the students working on the Lark grant, Zhang, presented a poster on Lark at the XSEDE 2013 conference. Brian Bockelman presented the Lark effort to the OSG Council, at a Fermilab Computing Division seminar, and at a faculty colloquium at UNL. Bockelman also presented the main goals of Lark in a CC-NIE breakout session at the Joint Techs TIP 2013 meeting.

What do you plan to do during the next reporting period to accomplish the goals?

During Year 2, we plan to disseminate the results achieved in Year 1 throughout the research community. In particular, we believe the Lark plugins for batch system network isolation is a novel approach to job virtualization and will write a paper describing how it works. Throughout Year 1, we have also been working to implement bandwidth management for jobs (currently only at the node level; WAN bandwidth management via the OpenFlow controller is a Year 2 goal). Because this bandwidth management isn't feasible without SDN, we believe this is another accomplishment worth writing up in a paper.

Year 2 should see the deployment of the Lark code into production. We will merge the Lark code into the UNL branch of HTCondor and deploy it to the CMS Tier-2 during the first six months of Year 2. The Lark team will enable the code on an increasing number of nodes at the CMS Tier-2 throughout the year - starting out with the NAT-based networking code. During the second half of Year 2, we predict we will have the Lark plugins committed into the main HTCondor repo. Once this is done and released as part of the official HTCondor sources, we will deploy the plugins at the UW Tier-2 site.

Building on the lower-level IPv6 work done on the UNL Tier-2 cluster in Year 1, we will switch the HTCondor install to IPv6 in Year 2. The current blocker for the switchover is the inability for the HTCondor daemons to respond to queries on both IPv4 and IPv6 -- this is important locally to UNL as IPv6 is not widely available across campus. We believe this will be delivered by the HTCondor team by December 2013. During Year 2, we also plan to extend the Lark HTCondor plugins to support acquiring IPv6 network addresses (from static configuration or SLAAC).

The HTCondor POX module will continue to develop throughout Year 2. The following improvements are expected:

- **Multi-switch support**: Extend the code to perform normal Layer-2 switching across several OpenFlow-enabled switches on the cluster, subject to the current network policy (split pool, with or without WAN, etc).
- **Cross-site flocking**: If we are able to construct virtual circuits between UW and UNL with DYNES, we would like to have the OpenFlow module isolate jobs coming from UW on a Layer-2 network connected directly with UW. If we are unable to create circuits with DYNES, we will investigate either GRE tunnels or a VPN.

- **Bandwidth management**: At the end of Year 1, UNL installed a new Brocade MLXe router to the datacenter core. The MLXe is able to perform traffic shaping at 10Gbps levels; we plan to extend the POX module to dynamically label each flow's traffic class based on the HTCondor job owner.

Finally, we want to work with the HTCondor team to start utilizing the collected perfSonar data. At the minimum, we plan on being able exclude each scheduler based on the amount of waiting data in the transfer if the estimated data transfer time is above a threshold (based on the perfSonar bandwidth performance estimates). Time permitting, we will also extend glideinWMS to not provision glideins for HTCondor schedulers whose estimated data transfer time is too long.