Gravitational-Wave Computing - IGWN Data Story

or Another Idiosyncratic Airing of Grievances, Condor-Style

Peter Couvares, LIGO Laboratory / Caltech

LIGO-G2301968-v1

Who am I?

- Peter Couvares — Staff Research Scientist at the LIGO Laboratory at Caltech
- LIGO Lab's Data Analysis Computing Manager — responsible for computing optimization (broadly defined) and success of distributed high-throughput computing (DHTC) for IGWN.
- Previously IGWN Computing & Software Committee Co-Chair 2015-2022.
- Before LIGO/IGWN, worked in industry (dataviz and telecom) and was part of the early HTCondor team at UW-Madison ('99-'09) and authored of the Distributed Acyclic Graph Manager (DAGMan) tool.

IGWN: <u>I</u>nternational <u>G</u>ravitational-<u>W</u>ave Observatory <u>N</u>etwork

- An umbrella designation to represent the common and coordinated infrastructure of <u>all</u> the second-generation gravitational-wave experiments worldwide: LIGO, Virgo, KAGRA, and LIGO India.
- Of practical benefit (igwn.org services don't highlight just one collaboration and don't all need to change names if/when we add another detector, unlike ligo.org or LVK)
- Important symbolically to emphasize our increasingly shared computing infrastructure and effort, and a potential precursor to a future, more formal, single overarching organization.

Why am I here?

- Because you asked!
- To share some of IGWN's operational issues and ask questions — you and IGWN face a variety of common research computing challenges:
    - Distributed computing challenges of large scientific collaborations.
    - Coordination of overlapping computing providers and distributed resource pools. Scientific workflow monitoring & operations.
    - The evolving intersection of High Performance Computing (HPC), High-Throughput Computing (HTC), and Cloud Computing.
- To share, learn from you, and find collaborators for future work…

Background: LVK Computing Scope Growth

- **SCALE** 🚀 - With ~1x the computing staff that we had in ~2009, we're supporting a ~5x larger user-base, delivering ~100x more computing power per observing run (10^6→10^9 CPU core hours) from ~10x the number of providers, while doing more science per computing cycle due to systematic optimization.
- **LOW LATENCY** - We support a complex, mission-critical, public-facing low-latency detection and alerting infrastructure on top of our previous almost-entirely batch/offline data analysis capabilities.
- **COLLABORATION** - We support seamless single-sign-on across almost every LVK service (numbering in the 100's) while providing many of the traditional productivity services of a large corporation (email/wiki/chat/document management/user database/etc.).
- **GITLAB** - We host a production git.ligo.org service to power everything from LVK software development, to automated testing, paper-writing, project management, and more.
- **IGWN** - We support the **unified** data analysis computing operations of the member collaborations, having replaced most parallel, duplicative, and incompatible infrastructure services with common IGWN solutions.
- **SHARED COMPUTING** - We transitioned from a labor-intensive "walled garden" of largely internal computing clusters to an open grid of shared resources into which we strive to efficiently incorporate CPUs or GPUs anywhere in the world without requiring data analysts to adapt to the peculiarities of individual providers.

Computing & Software

- We want to **rely on external providers** when it make sense. E.g., for identity and access management (CoManage, InCommon, etc.), distributed computing infrastructure (HTCondor, OSG), etc. — experts who can do things better than we can in-house, and on a larger scale.
- We need to stop making things more difficult than they need to be. **Less customization, fewer proprietary requirements**, etc. There is a large hidden computing cost for anything non-standard. The more we can adapt to match what HEP and others do, the better. (And better yet if we plan together in advance…)
- IGWN has seen enormous benefit in relying on **trusted liaisons** and intermediaries (e.g., PATh) to interface with disparate service providers — especially when share those providers in common with other large projects.
- We'd like to **avoid duplication of effort** and leverage solutions others have developed.

One Success Story: Condor/OSG/ PATh

- LIGO adopted a High-Throughput Computing model over 15 years ago, and the HTCondor ecosystem (HTCondor, DAGMan, etc.) is in the core of IGWN's computing DNA. IGWN's relationship with OSG is anchored on this pre-existing culture, computing model, and software stack.

- OSG provides a "**universal adapter"** to diverse resource types: dedicated internal resources, major shared computing centers (e.g., CNAF, IN2P3, Nikhef), large campus clusters, opportunistic CPUs, supercomputer allocations (e.g., XSEDE/ACCESS, Compute Canada, etc.), and commercial cloud CPUs.

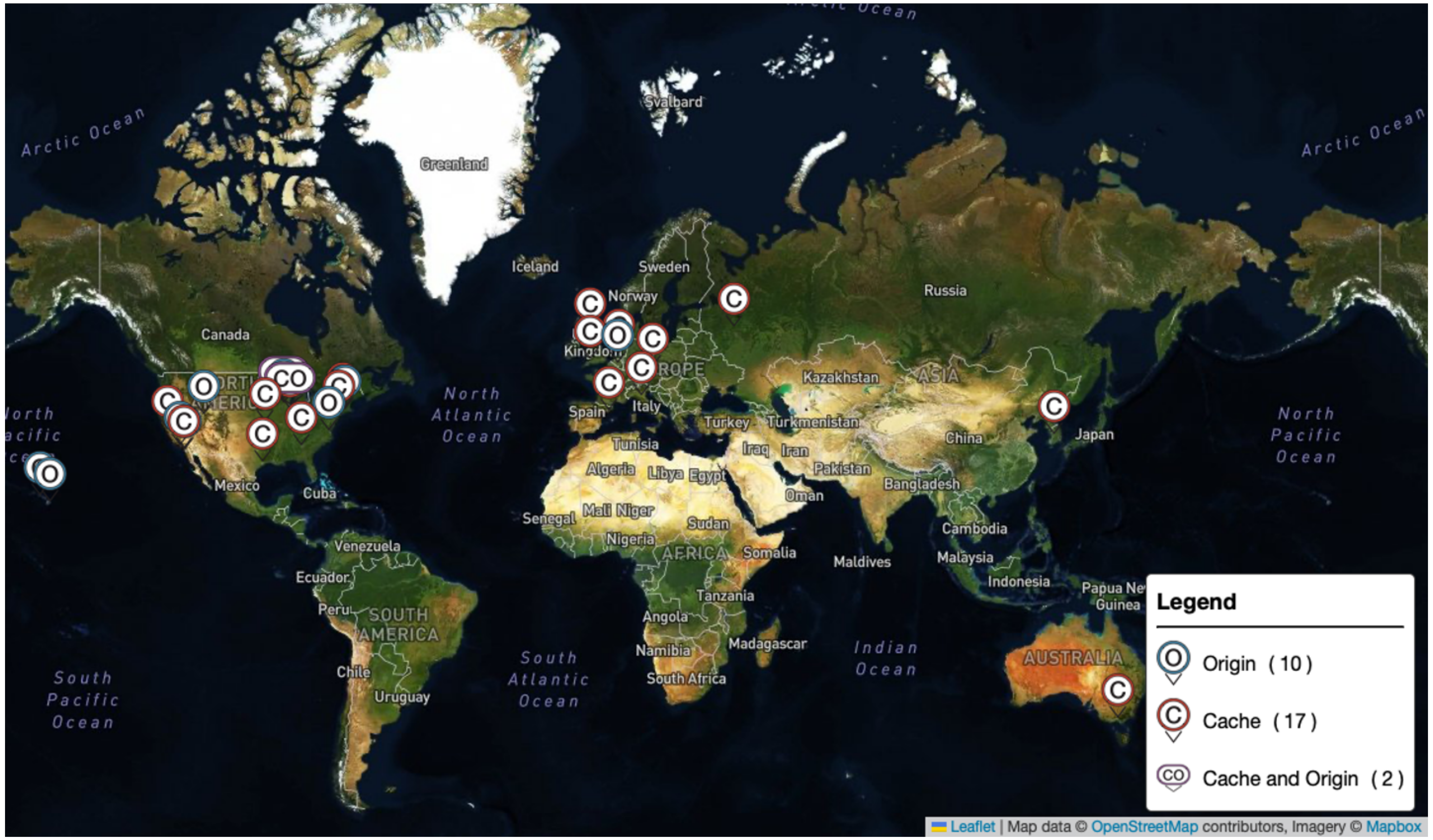
- OSG **hosted services** (GlideinWMS, hosted CEs, etc.) + expert support = production infrastructure without a huge IGWN-specific investment.

- **Track record of success** in HEP — High Energy Physics (LHC) computing forged a path, thanks to the work of many of the people in this room.

- Friendly, enthusiastic, skilled, **results-oriented**, **flexible** PATh management and staff. Not hung up on boundaries, processes — focused on science goals first.

Data and OSDF

- IGWN is making increasingly heavy use of the worldwide Open Science Data Federation (OSDF) network, including ESnet caches at London & Amsterdam POPs — via CVMFS, HTCondor file xfer, and native OSDF/ Pelican.


Legend
Origin (10)
Cache (17)
Cache and Origin (2)
Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox

It’s Not Easy Getting Data to Jobs

- job/workflow input data (science data, parameters, configuration)
  - pushing everything from the submit node is easy for users but doesn’t scale…
  - CVMFS is easy for users but has never been reliable for authenticated data…
  - OSDF is promising but still immature…
- Increasingly distributed & dynamic CPU/GPU resource pools — giant (10k+ job) workflows run on different resources (sites) each time you run them.

This is What I *Hoped* to Present…

"Results from My Job-Routing Experiment to Shorten Workflow Tails!"

- We have large IGWN workflows (10k-100k jobs in a DAG) that often take a week or more of wall-clock time to complete.
- For the sake of this talk, a simplified model workflow consists of 10k independent jobs, each of which requires a different ~1GB of input data (stored on the local AP filesystem, in CVFMS, *and* in OSDF — user can take their pick), runs for 4 hours, and then transfers 1 MB back to the AP (via Condor file xfer).
- Time profiles of these workflows show that most of the jobs complete in the first day, and the remaining fraction take a week or more to finish. Why do 10% of the jobs take 90% of the wall-clock time?
- Because jobs fail and re-run, fail and re-run again, etc.
- **Lots of different root causes** — very hard to prevent/solve all of them. ("Whack-a-mole", "List of doom", etc.)
- **But**: at any given time, failures tend to be concentrated at certain sites (sometimes a dozen or more sites — not always just one or two). Jobs, like lemmings, keep running off the same cliffs.
- There are usually a few sites that seem to have very few failures. Let's call these "first-class sites".
- What if we could do the simplest thing possible? How about: tell Condor to run jobs anywhere and everywhere — but once a given job fails once, for any reason, only re-run at a "first-class" site.
- There are obviously much cleverer things we can do, but this is simple and should be a good test of whether better job routing can shorten our long tails meaningfully.
- Run some "control" workflows (with no special routing) alongside some test workflows with special routing, and measure the improvement, if any.

What Happened Instead

- Many months later… and I still can’t reliably run workflows to completion!
- As an increasingly pointy-haired person, it had been a while since I had run a “real” LIGO workflow. I hadn’t run anything this big since we had fully migrated to OSDF, tokens, etc.
- …and the reasons for problems were so inscrutable. Why am I getting so many shadow exceptions? Why are the jobs going on hold? Should I release them? Remove/resubmit?
- I knew our IGWN Pool user support expert (James Clark) was overwhelmed, but I hadn’t fully appreciated why, or the extent to which he was the only person who understood the error messages or how to work around them (in some cases this included the Condor team).
- If I couldn’t figure this out, what hope could a normal ~~human~~ IGWN scientist without decades of Condor experience have? Most people just do “while(1) condor_release -a && sleep 600”.
- I became totally sidetracked by a new mission: understand what’s going on, and push the Condor team to improve the “user experience” for large workflows facing multiple errors.
- (Also, push the Condor team to recover and expose fewer errors, period.)

Problems with Data Transfer -> Usability Nightmare

- Condor file xfer, CVMFS, OSDF — none worked well.
- Timeouts
  - slow/overloaded network to specific CE/s, slow/overloaded network from AP, slow/overloaded network from OSDF cache, EP preemption
- Auth
  - Auth failures due to EP misconfiguration (CVMFS failing due to wrong x509 CA certs), Auth failures due to user error (e.g., sent along wrong kind of x509 cert - delegated vs. whatever, sent along wrong Scitoken), Auth failures due to OSDF/token problems (bugs)
- CVMFS
  - fopen/read errors (many causes)
- OSDF
  - caches missing data they should have had, etc., wrong cache chosen, etc.
- other causes — too many to list
- **Almost all of these are completely impossible for the end user to understand or differentiate.**
- Until recently, almost all of these resulted in shadow exceptions, then retries, then job holds.

It’s Too Hard to Debug Failures from the Access Point (AP)

- For the User
  - Jobs match and restart for no discernible reason.
  - Jobs go on hold with opaque errors.
  - Jobs appear to be running but make no progress.
- For the Admin
  - Pilot failures are particularly hard to debug — key information doesn’t make it back to the AP.
  - Preemption by the batch system behind the CE is particularly hard to identify — key information doesn’t make it back to the AP.
  - Job Classad, Job Userlog, Condor ShadowLog are only “official” channels for information — insufficient.
  - What turns out to be needed: pilot logs, CE logs, front-end/factory logs, etc.
- Caveat: some failures are a consequence of our own sins: e.g., CVMFS issues. What Condor doesn't manage as a first-class operation, Condor can only do so much to help you debug. But error/information propagation from is still very poor.

Next Steps - More Complicated Data Mgmt

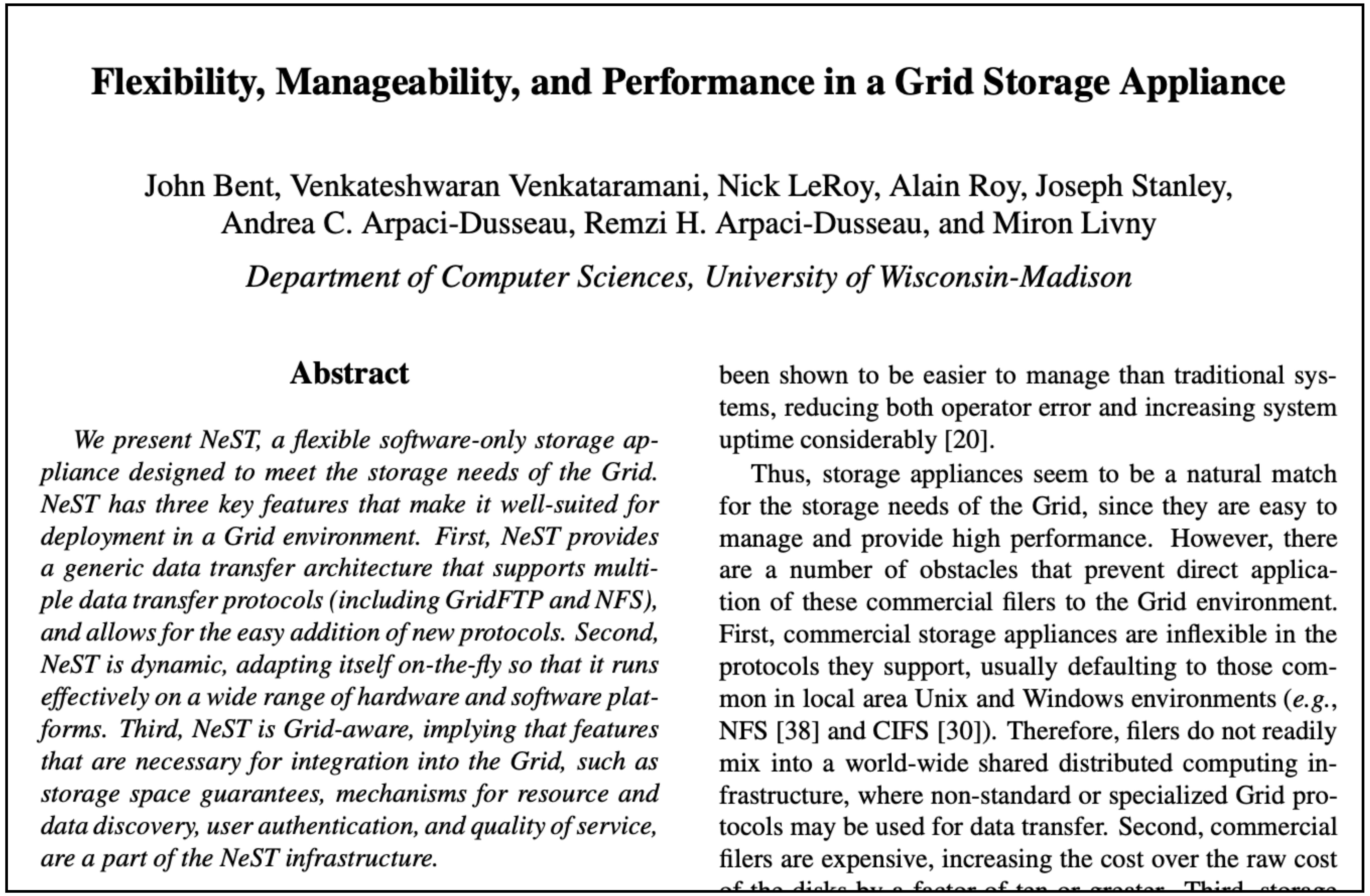
- running job inspection, debugging, progress monitoring
  - in a local pool, shared submit/execute filesystems and condor_ssh_to_job work well
  - neither work well on distributed pools (OS Pool, IGWN Pool, etc.)
  - makes no sense to always xfer/stream data back when you only need it once in a blue moon
  - this is a major practical usability regression for users moving from local pools to the grid (OS Pool, IGWN Grid, etc.)
- job checkpoint data
  - used to be Condor's problem to manage, now it's each users' problem — oh for the days of yore!
- intermediate workflow data (inter-job results or inter-workflow data movement)
  - back-and-forth to the submit node is state of the art for most LIGO-Virgo-KAGRA workflows
- workflow output data (science output, workflow metadata, debugging info)
  - back to the submit node is state of the art for most LIGO-Virgo-KAGRA workflows
- FUTURE: runtime telemetry (scientific and computing progress metadata)
  - should we do this /around/ Condor, in coordination with Condor (?), or via Condor (e.g., classads, chirp)?

Questions

- IGWN is taking more responsibility for troubleshooting problems for our users.
  - Jobs match and restart for no discernible reason. Jobs go on hold with opaque errors. Jobs appear to be running but make no progress.
- We're increasingly struggling with CE/provider monitoring and operations
  - Knowing when sites are offline or otherwise unable to run jobs successfully, and responding once we know. (Pilot failures are hard to debug — key information doesn't make it back to the Access Point.)
  - Validating data access (e.g., authenticated CVMFS) before a job runs.
- **Can/should we make better use of work that's already been done to monitor sites, validate resources immediately before jobs run, and/or respond to problems?**

Conclusion - More Questions

- **What are the right data models, storage solutions, and auth/xfer implementations to simplify data movement for users?**
- I don't know yet…
- Over 20 years after the June 2002 Condor NeST paper, we still don't have DHTC storage discovery, authentication, authorization, allocation, QoS, timeout/cleanup, etc.
- But many promising building blocks are here (SciTokens, OSDF, Rucio, etc.). Can we pull them together?



Flexibility, Manageability, and Performance in a Grid Storage Appliance

John Bent, Venkateshwaran Venkataramani, Nick LeRoy, Alain Roy, Joseph Stanley, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Miron Livny

Department of Computer Sciences, University of Wisconsin-Madison

Abstract

We present NeST, a flexible software-only storage appliance designed to meet the storage needs of the Grid. NeST has three key features that make it well-suited for deployment in a Grid environment. First, NeST provides a generic data transfer architecture that supports multiple data transfer protocols (including GridFTP and NFS), and allows for the easy addition of new protocols. Second, NeST is dynamic, adapting itself on-the-fly so that it runs effectively on a wide range of hardware and software platforms. Third, NeST is Grid-aware, implying that features that are necessary for integration into the Grid, such as storage space guarantees, mechanisms for resource and data discovery, user authentication, and quality of service, are a part of the NeST infrastructure.

been shown to be easier to manage than traditional systems, reducing both operator error and increasing system uptime considerably [20].

Thus, storage appliances seem to be a natural match for the storage needs of the Grid, since they are easy to manage and provide high performance. However, there are a number of obstacles that prevent direct application of these commercial filers to the Grid environment. First, commercial storage appliances are inflexible in the protocols they support, usually defaulting to those common in local area Unix and Windows environments (*e.g.*, NFS [38] and CIFS [30]). Therefore, filers do not readily mix into a world-wide shared distributed computing infrastructure, where non-standard or specialized Grid protocols may be used for data transfer. Second, commercial filers are expensive, increasing the cost over the raw cost