Journal of Computational Science xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Principles, technologies, and time: The translational journey of the HTCondor-CE

Brian Bockelman [a,*], Miron Livny [a,b], Brian Lin [b], Francesco Prelz [c]

[a] Morgridge Institute for Research, Madison, USA
[b] Department of Computer Sciences, University of Wisconsin-Madison, Madison, USA
[c] INFN Milan, Milan, Italy

ARTICLE INFO

Keywords:
Distributed high throughput computing
High throughput computing
Translational computing
Distributed computing

ABSTRACT

Mechanisms for remote execution of computational tasks enable a distributed system to effectively utilize all available resources. This ability is essential to attaining the objectives of high availability, system reliability, and graceful degradation and directly contribute to flexibility, adaptability, and incremental growth. As part of a national fabric of Distributed High Throughput Computing (dHTC) services, remote execution is a cornerstone of the Open Science Grid (OSG) Compute Federation. Most of the organizations that harness the computing capacity provided by the OSG also deploy HTCondor pools on resources acquired from the OSG. The HTCondor Compute Entrypoint (CE) facilitates the remote acquisition of resources by all organizations. The HTCondor-CE is the product of a most recent translational cycle that is part of a multidecade translational process. The process is rooted in a partnership, between members of the High Energy Physics community and computer scientists, that evolved over three decades and involved testing and evaluation with active users and production infrastructures. Through several translational cycles that involved researchers from different organizations and continents, principles, ideas, frameworks and technologies were translated into a widely adopted software artifact that isresponsible for provisioning of approximately 9 million core hours per day across 170 endpoints.

1. Introduction

In 2014, the Open Science Grid (OSG) consortium announced to the sites in its Compute Federation that it would base its computing resource management technologies on the HTCondor Compute Entrypoint (CE) [1]. The HTCondor-CE enables remote submissions of acquisition requests to a compute resource. Developed and maintained on top of technologies from the UW-Madison Center for High Throughput Computing (CHTC), this edge service replaced the Globus Gatekeeper adopted by the OSG upon its inception in 2005. The announcement was a major milestone in a project that began two years earlier within the OSG with roots that extended into more than a decade of collaboration with multi-disciplinary projects in Europe, two decades of collaboration between Computer Science and High Energy Physics (HEP), and more than three decades of distributed computing research and experiences. Over time we witnessed processes of translational [2] nature that played a key role in reaching this milestone. Today, more than 170 endpoints at more than 65 sites worldwide use the HTCondor-CE to facilitate sharing of computing resources via remote acquisition.

Like other elements of the evolving HTCondor Software Suite (HTCSS) [3], the HTCondor-CE project has a starting date but has no end date in sight. In the context of our work, successful translational projects start by the articulation of the need for a capability by a committed customer and continue as long as the software artifact is deployed. The commitment that triggered the current translational cycle came with a 2012 assignment by the OSG Executive Team to its "Technology Investigations" activity: under the leadership of Bockelman and in collaboration with CHTC, develop a new strategy for the OSG Compute Entrypoint. The very nature of a software artifact that provides a critical capability in a cyberinfrastructure ecosystem requires a commitment to long term evolution founded on mutual trust and sustainability. The long partnership between OSG and CHTC has been facilitating such a commitment.

The HTCondor-CE embodies principles, ideas and technologies that have been pioneered and developed initially by the Condor Project that started in 1984 [4] and more recently by the CHTC. As a HTCSS element, the HTCondor-CE leverages software components and partnerships that were developed over three decades by the group led by Livny. The

* Corresponding author at: Morgridge Institute for Research, 330 N Orchard Street, Madison, WI 53715, USA.
E-mail address: bbockelman@morgridge.org (B. Bockelman).

https://doi.org/10.1016/j.jocs.2020.101213
Received 31 May 2020; Received in revised form 26 July 2020; Accepted 25 August 2020
Available online 20 September 2020

hallmark of these partnerships has been the translation of advances in distributed computing into scientific discovery through an open High Throughput Computing (HTC) software ecosystem.

In our experience, the roots of translational projects are typically diverse and run deep. While recent changes in the technology landscape have impacted the evolution of the HTCondor-CE, the abstractions presented and the architecture of the software have been established for decades. The HTCondor-CE includes elements of the HTCSS that were first introduced two decades ago, such as the ClassAds language to describe acquisition requests, and resources, and matchmaking [5] to route requests to resources. As in the case of Translational Medicine, the software delivered by a translational computing project has to be of value and of low risk to the target community. Namely, the software must be dependable, reliable and secure. Achieving and sustaining these goals requires creative software reuse, adaptation of new technologies and disciplined software engineering.

The community served by the OSG is international with a significant involvement of the Large Hadron Collider (LHC) experiments at CERN [6]. This entails a close relationship with the Worldwide LHC Computing Grid (WLCG) [7]. In 2005, the OSG deployed the gatekeeper software widely adopted by the WLCG. The WLCG would later decide to replace its gatekeeper. The search for an alternate technology that was conducted by the European gLite project [8] offered the Condor Project an opportunity to develop a prototype that would underpin many of the later HTCondor-CE technologies. In 2007 the gLite project eventually selected a different technology for its gatekeeper (CREAM [9]), bringing the previous translation cycle to a halt. At the time the OSG decided not to follow the WLCG.

The HTCondor-CE project builds on a close and productive partnership between the CHTC and the HEP community. It started in 1990 with researchers from Nikhef in the Netherlands [10] and continued in 1995 with researchers from INFN in Italy [11]. A key element of the HTCondor-CE has been developed and maintained by the INFN and the concept of "flocking" across HTCondor pools was developed through the Nikhef collaboration. In the early 2000s, these partnerships expanded to additional organizations and projects and were formalized through three US projects (PPDG, GriPhyN and iVDGL) that led to Grid3 [12] and the OSG and through two European projects led by CERN (European DataGrid and EGEE). This groundwork helped develop capabilities that serves the entire spectrum of open science domains today. A number of WLCG sites in Europe (including CERN) and Asia have adopted the HTCondor-CE.

The paper provides an overview of the motivation, functionality and architecture of the HTCondor-CE and presents the history and elements of the translational project that took a problem statement articulated by a customer and delivered a widely deployed software tool. The project is now eight years old with a history that goes back almost four decades.

2. Background and context

The application of the translational principles embodied in the HTCondor-CE project started with the realization, in the late 1990s, that the scale and complexity of the data-taking for the LHC exceeded the capabilities of CERN in-house compute resources, of physicist-driven software development that had carried the HEP field on for decades, and of available commercial solutions [13–16]. Rather than scale back the physics goals, the LHC reached out to Computer Science groups, finding the nascent idea of wide area distributed computing promoted in the context of "grid computing" [17], a neat fit to allow the LHC experiments to implement a multi-tier infrastructure that leveraged resources at sites around the world in addition to the resources at CERN. Each of these labs and universities provided their computational resources through a batch system, such as HTCondor, LSF, or PBS, that users would access and submit computational jobs through a local login host via SSH. When faced with the need to share these resources with a globally-distributed user community, the initial approach – based on the model developed by the Globus project [18] – was to provide a mechanism to submit these jobs remotely. The software that enabled remote job submission was termed the gatekeeper [18]. A compute cluster accessible via such a remote job submission gateway was considered a WLCG Compute Element or, in short, a CE.

The paradigm of remote job submission was not particularly successful: it was frustrating to end-users when a compute element would lose jobs or when jobs were sent to a site where they got "stuck" in a remote queue waiting to be selected by the local scheduler. This approach of "early binding" hinged on reliably predicting the behavior of job queues, which proved unreliable. Error propagation across the software layers was difficult. These pitfalls of a job-centric computing model are covered by Sfiligoi [19].

2.1. The OSG compute federation

The OSG gatekeeper services consequently shifted from managing jobs to acting as a resource acquisition service. The "CE", in its new function as "Compute Entrypoint," securely manages various local resources: clusters where access is granted by a batch system, VMs managed by "cloud" services, or other CEs.

In the OSG ecosystem, the remote clients of CEs are no longer end-users but rather "factories" that represent organizations responsible for managing the jobs submitted by end-users; rather than end-user jobs, the factory service sends to the CE Resource Acquisition Requests (RARs), commonly referred to as "pilots." Once a resource is allocated by the Local Resource Management System (LRMS), a pilot process is launched and joins a pool of resources managed by the organization that operates the factory. The CE can be viewed as a service that routes RARs between the factory and the service managing the local hardware.

A resource management overlay is established by the organizations that manages the single logical pool of acquired resources, assigning them end-user jobs ("payloads"), with tight control over reporting and error propagation. Further, since payloads are not committed to a resource until it is available (late binding) and executed by the deployed pilot, users are not exposed to the scheduling vagaries and the APIs of dozens of autonomously managed LRMS's.

GlideinWMS is used to provide factory services to most OSG organizations [20]; in GlideinWMS, the HTCondor scheduler - using "HTCondor-G" – submits and manages RARs at remote CEs [21]. The scheduler is also used to manage the deployed pool of resources. Upon start-up, the pilot downloads, configures, and launches an HTCondor execute node which securely joins the HTCondor pool managed by the organization. At sites where HTCondor is also the LRMS, OSG leverages the fact that there is a consistent software suite from factory to CE to LRMS to resource pool, minimizing information loss due to 'translation' between the layer as each layer communicates using the ClassAd language.

2.2. Functionality needed by a CE

To demonstrate the paths between desired functionality and basic research we highlight four core CE functions required by the OSG along with the corresponding basic technology research:

1 **Manage Resource Acquisition Requests**: The CE has to track RARs throughout their lifecycle while communicating pilot, status, accounting information, error information, and log files back to the factory.

Research: For HTCondor-G, the HTCSS scheduler (the "SchedD") is used to manage the RARs (each request is represented by a job in the SchedD) even though RARs are fulfilled by an external system. This allows the same interface to be used regardless of whether the entity represents a job executed by HTCSS itself or another LRMS. Work on the architecture of HTCondor-G dates back to 2001 [21]. This is enabled by

the semi-structured properties of the ClassAd language that is used to represent all entities throughout the HTCSS.

2 **Interacting with the LRMS**. HTCondor-G was originally designed to execute a job in a remote batch system using the Globus Gatekeeper. However, the design evolved to be highly modular so new modules could be written for other services providing a job-like interface. For example, since the lifecycle of a virtual machine in a cloud system goes through states that can be mapped to batch job states (idle/pending, running, removing), the HTCSS scheduler can manage virtual machines in multiple commercial cloud providers.

Research: Primarily, the resources provided by OSG sites are managed by a variety of batch systems. Matching interfaces have to be implemented, tested and maintained. The BLAHP [22], software originally developed as part of both the gLite CE [23] prototype and CREAM services (see Section 3) is an example of architecture and protocol for this purpose.

3 **Translate from remote RAR to local RAR**: The factory should not have detailed knowledge about the local resources. Rather, it is important that RARs express generic requirements and are only augmented with local information once at the site. This facilitates the autonomy of the local site administrator.

Research: The HTCSS developed the concept of a JobRouter in 2009. It is based on a service that passively monitors the queued jobs and can take an incoming job, place it on hold, follow a specified set of transformation rules encoded in the ClassAd language to create a local job, and submit the derived job back into a SchedD for local execution [24].

4 **Externally authenticate and authorize actions**: The CE ensures the resource provider the incoming RARs are appropriately authorized. This is most commonly done by authenticating the remote entity to a global identity, mapping this to a local identity, and then applying a set of authorization policies based on that local identity.

Research: The CEDAR communication framework [3] developed by HTCSS is based on protocols which negotiate an authentication mechanism with a remote entity, establish an identity, and apply a set of rules to determine whether the identity is authorized for an action. As the authentication protocol itself is negotiated, CEDAR has the ability to evolve; while GSI [25] is currently the most popular protocol within the OSG, CEDAR can utilize different and evolving technologies such as the capability-based SciTokens [26].

2.3. The HTCondor-CE architecture

Four elements of the HTCSS compose the HTCondor-CE as illustrated in the flow of a RAR from the factory HTCondor-G to a HTCondor-CE to the LRMS in Fig. 1. The elements were enhanced in the course of this translational cycle to meet specific needs of a CE service and are now standard features of the software suite. The HTCondor-CE meets the requirements set forth within Section 2.2 solely through a special configuration of these elements. The HTCondor-CE consists of the following elements:

- **SchedD**: The SchedD authenticates the remote client and accepts the incoming RAR **R**. These requests are managed as job entities in the internal SchedD database.
- **JobRouter**: The JobRouter takes the generic RAR (**R**) directly from the SchedD database that is logged to disk and customizes it to make an equivalent local RAR (**B**) and submits it back to the SchedD. The JobRouter is responsible for ensuring updates to **R** and **B** follow a two-way mirroring protocol. For example, the JobRouter will update **R**'s state to running when **B** is marked as running; if the remote client removes **R**, the JobRouter will remove **B**.
- **BLAHP**: The RAR **B** is represented as a job in the HTCondor's "grid universe". In the SchedD, this indicates the job is to be managed by an external entity — in this case, the **BLAHP**. The **BLAHP** is given the description of **B** and is responsible to convert this description into a job **J** for the LRMS; the SchedD will also invoke the BLAHP to perform status updates and job removals. Currently supported LRMSs are Grid Engine, HTCondor, LSF, PBS Pro/Torque, and Slurm.
- **Collector**: In the case of GlideinWMS pilots, the collector aggregates information from the running pilots and forwards it to the OSG information service (also a HTCSS collector). This includes CE contact details (allowing for service discovery), a summary of the CE configuration and RAR monitoring in the HTCondor-CE.

3. Translation process

As noted, much of the translational activity was driven by collaboration between CS and HEP. To borrow from a different definition of the word translation, the collaboration between the two disciplines required projects to "translate one's own disciplinary jargon into a language that can be understood by others." [27] It also required that the outcomes and artifacts produced by the CS research possess rich enough semantic

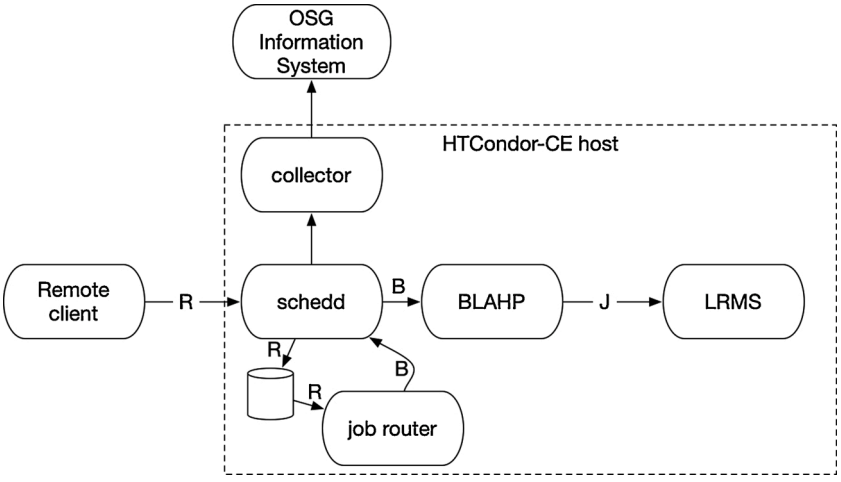


Fig. 1. The architectural components of the HTCondor-CE. Here, R represents the incoming RAR from the remote client. This is transformed according to policy into a local "blah" job, B. The SchedD also tracks B's lifecycle in its internal database and uses the blah to interact with the LRMS. The blah will take B, converted to an appropriate batch job J, and submit it to the LRMS.

properties to allow restructuring or re-combining for use in new and shifting contexts. This led to the identification of gaps in abstractions then to the development of bridges and connections between the collaborating parties.

Throughout the early 2000's, the HTCondor project developed the frameworks and technologies that would eventually underpin the HTCondor-CE. While the individual components were developed to advance distinct research goals, all were driven by a vision of enabling Distributed High Throughput Computing on a global scale. The move from the lab (the CHTC at the UW–Madison campus) to the locale (initial deployments on OSG) to the community (widespread production deployments) would span 12 years. This includes an initial failed translation cycle, international collaboration, and the initial adoption by leaders in the community. The timeline – from initial research through community adoption is summarized in Fig. 2.

3.1. From lab to the locale

A first attempt at developing a CE based on the distributed computing principles underpinning HTCondor's basic research was the 'gLite CE' [28] prototype. It was an early attempt to translate the abstractions and technologies of the HTCSS into a functioning CE that could be deployed by an organization. In the gLite CE architecture, the site gatekeeper service (the Globus Gatekeeper software) would launch on behalf of an organization an unprivileged version of the HTCSS SchedD. In this way, each LHC experiment in the WLCG could utilize a bespoke CE, customized for their needs, policies, and potentially protocols. The SchedD process, specific to the remote organization, would then forward the jobs to the LRMS using the newly developed BLAHP. This approach offered more autonomy to the organization in how they used the remote resources. These spheres of autonomy are critical to the effectiveness of a distributed system – but come at a cost of complexity. Contrasting with Fig. 1, there would be multiple SchedD's on the LRMS head node, the early prototype did not include a JobRouter to customize jobs (it was assumed the external submitter would customize based on the available global information in discovery services [29]), and the CE service was still job-oriented rather than RAR-oriented. Had the WLCG followed this approach, organizations could have deployed RAR-oriented CEs as they adopted the pilot concept.

The prototype development effort stemmed from the gLite project's desire to replace the Globus gatekeeper as the service provider and sole port of entry for the remote resource. During this development effort, the gLite project investigated two distinct technologies: the gLite CE and the CREAM CE [30] developed by INFN. After an internal evaluation of both technologies, in 2007, gLite decided to adopt the CREAM CE. While we were not privy to the decision-making process, we believe this highlights the fact the customer makes the decision – and can include considerations beyond simple technical items. This fact that adoption is a complex, sometimes non-technical, process is a major obstacle in translational work.

At the time, OSG decided to stay with Globus Gatekeeper and adopt neither of the other CEs – the need to transition from Globus was not seen as urgent and there was no desire to undertake a software engineering project at the time. Given neither the OSG nor gLite decided to adopt the gLite CE approach, this translation was essentially a failure. Despite the failure to reach the target community, a positive outcome of this era was further development of the software (HTCSS, BLAHP) that would eventually underpin the HTCondor-CE. For example, the BLAHP was integrated as part of HTCSS releases and HTCondor gained significant capabilities to accept jobs through remote submission. HTCSS was enhanced with the HTCondor-C capability that supports remote submission between two SchedDs

In parallel to the gLite CE prototype work, the HTCondor project developed the concept of a "SchedD on the side", which acts like a "shadow" of a SchedD, creating a new job that is derived from and linked to the original job in the SchedD after a specified transformation. This approach leverages the HTCSS view of a job as a chain of job instances anchored by the original job submitted by the end-user and dynamically expanding and shrinking through delegation to other services. The concept also leverages the log-based technology used by the SchedD to

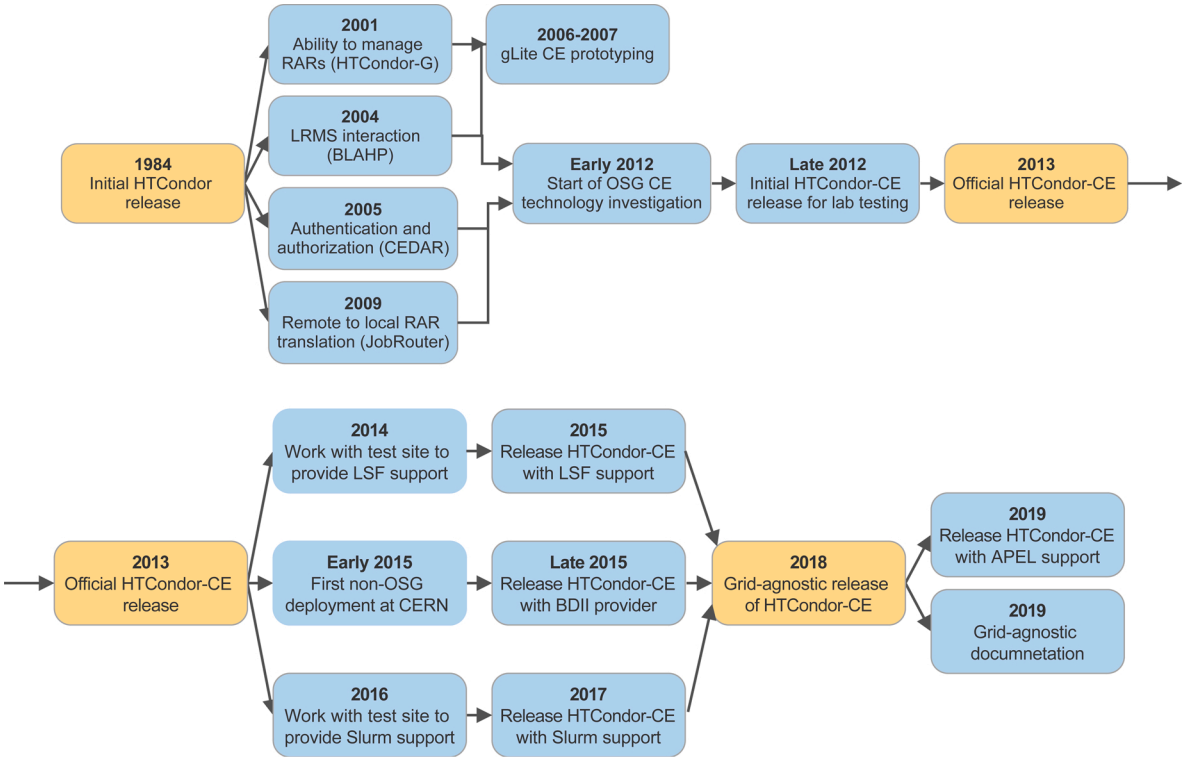


Fig. 2. A timeline of major events in the lifetime of the HTCondor-CE, from initial work on HTCSS, to components preceding the HTCondor-CE, to details from the CE's translation.

manage a disk image of the in-memory set of job ClassAds. Trailing the transaction log enables the JobRouter to shadow the current SchedD state and keep an up to date, in-memory image of the collection of jobs managed by the shadowed SchedD service.

During this period, the gLite project would continue to integrate the CREAM CE and OSG would base its CE product on software from Globus. For their gatekeeper, Globus would adopt a Java-based SOAP web service-based approach for GRAM4 [31] before ultimately abandoning this approach for an evolution of its original software stack in GRAM5 [32].

The next attempt to translate HTCSS as a CE for the Distributed High Throughput Community began in 2012, when the OSG decided to investigate alternatives to the Globus Gatekeeper; this new investigation was triggered by uncertainty in the future support for the Globus Gatekeeper component and the opportunities afforded by the earlier transition to the resource acquisition model. Over the next 6 months, the OSG internally evaluated two options: the HTCondor-CE, based upon HTCSS and developed by Bockelman, or the CREAM CE. Given HTCSS was already a central component of many existing pieces in the OSG technology stack — including the GlideinWMS resource management overlay — the HTCondor-CE (as described in Section 2.3) was built from familiar technologies and did not add dependencies on new external software providers.

While the HTCondor-CE has several overlapping ideas with the gLite CE, it was a completely new project, initially sharing no developers or code. Compared to the prior gLite CE, the HTCondor-CE prototype was simpler — as the HTCondor SchedD was running as a privileged executable, this process could serve all clients of the CE, regardless of the mapped Unix user of the client. The HTCondor-CE would also perform the authentication and authorization of the remote entity through invoking the Globus GSI libraries as opposed to relying on a separate gatekeeper process to authenticate clients prior to HTCondor starting. Finally, by using Globus GSI for authentication and LCMAPS [33] for authorization callouts, the CE would rely on the same infrastructure as the rest of the community, helping to gain acceptance. The HTCondor-CE is seen as a highly customized configuration of HTCSS as opposed to a major software engineering project by itself.

The initial release of HTCondor-CE was done in May of 2012 and showed functionality for two LRMS's (HTCondor and PBS) — while functional, it was still in the 'laboratory'. This initial release was hardened in the OSG Integration Test Bed (ITB), which allowed for internal testing and integration of HTCondor-CE with the rest of the OSG Software Stack. After completion of testing in this "laboratory", HTCondor-CE was released to the OSG in its November 2013 production release. The first production deployment was in 2013 at Nebraska; another early adopter was Brookhaven National Laboratory (BNL) in 2014. After these initial successes, the OSG decided to officially transition to using the HTCondor-CE as the base for its OSG CE product; at the time, both HTCondor-CE and GRAM were supported as "backends".

The two largest resource providers in the OSG Compute Federation are the U.S. ATLAS and U.S. CMS operations programs; their adoption of the new software was both a critical test of its functionality and a vote of confidence in the technology. The U.S. CMS operations program started their transition to the HTCondor-CE in April 2014 while the engagement with U.S. ATLAS began in June 2014. The work with ATLAS benefited from a contributor who was both a member of the OSG release team and managed the U.S. ATLAS CEs at BNL (deploying the HTCondor-CE in August 2014). Other sites helped expand the functionality as additional LRMS's were integrated; for example, SLAC helped contribute and test the LSF support and corresponding documentation starting June 2014; SLAC eventually moved their endpoint to production in March 2015. Similarly, native Slurm [34] support would occur over the next year. In August 2015, the OSG announced that it would drop support for Globus GRAM completely in the following year, beginning HTCondor-CE's transition from the locale to the community [35].

3.2. From locale to the community

We consider the OSG decision in 2015 to base its computing software on HTCondor-CE as the first indicator that the software begun its transition to the community — the software was increasingly critical to CHTC stakeholders and provided core functionality to the OSG. A second important milestone for the HTCondor-CE was its adoption by CERN. In 2015, CERN was in the process of adopting HTCondor as a LRMS and evaluating new CE technologies. Eventually, CERN decided to also use HTCondor-CE — largely based on the strength of the integration with other components of HTCSS and that it would only depend on a single software provider (CHTC) for both the LRMS and the CE. While HTCSS has been used in the WLCG beyond the OSG, the CERN endpoints were the first non-OSG deployment of the HTCondor-CE.

Once proven in the locale, the core architectural components of the HTCondor-CE have remained relatively fixed. As usage of the CE spread, additional integrations needed to be performed in order for the software to work in new communities. For example, while the native information service mechanism is the HTCSS collector (as shown in Fig. 1), in Europe the BDII is used for service discovery [29] and support had to be contributed to the HTCondor-CE software. Other differences at European sites included integration of the new CE with the local accounting software (APEL [36]) and authorization service (Argus [37]).

While the HTCondor-CE derives from the widely used HTCSS and uses the same authentication and authorization libraries, it is still a complex service exposed to the internet; several sites involved in this transition process expressed security concerns. To gain wider acceptance in the community, the HTCondor team solicited review by the Center for Trustworthy Software Cyberinfrastructure / Trusted CI [38] in August 2016. The review was completed in August 2018 with only minimal flaws for the CHTC team to resolve.

Another hallmark of software broadly used in the community is the provisioning of a multiple-channel support structure. HTCondor-CE support was provided initially through the existing OSG ticketing systems and, as the software moved beyond the OSG locale, the HTCondor mailing lists and ticketing systems. Beyond simple support, the OSG also performs outreach in terms of presentations and tutorials at a number of forums, including HEPiX 2015 [39], HTCondor Week Europe 2016–2019 [40–43], OSG All Hands Meetings (starting in 2013), and Tutorials (e.g. at ISGC 2019 [44]).

Structural changes were needed as the scope expanded further beyond the OSG. Several internal parameters (access credentials, accounting methods, configuration, management software) made implicit assumptions about being run on the OSG CE. Support of non-OSG sites first became viable with the 2018 'grid-agnostic' version of HTCondor-CE. Any implicit OSG assumption was moved to a separate package - non-OSG sites only needed the base package. This work was completed in 2019 when corresponding OSG-free documentation was released along with moving the code from the OSG GitHub organization to HTCSS.

4. Impact and lessons learned

The HTCondor-CE has become a major service of the OSG and a project within the HTCSS. It is the mechanism through which resources are delivered to communities and is relied upon daily. While not all CEs need to be publicly advertised, at the time of writing 174 endpoints (each representing typically a large cluster at a university or lab) are known; of these, about 90 % can be queried from the public Internet. Over the past two years, the number of deployed endpoints has grown by about 100.

On a typical day, these queryable endpoints manage 875,000 pilots (this includes running RARs and those pending). The typical resource acquisition per RAR varies (making it difficult to estimate the total cores served) but the accounting systems at OSG and CERN each record over 160,000 cores utilized by pilots on average; the total number of cores

shared through HTCondor-CE endpoints is likely over 400,000. Whether measured by cores or through the breadth of science enabled, the translation of the HTCondor-CE has made major impacts in this community.

Achieving this impact allows us to reflect on some lessons learned. First and foremost, **translational computing requires real time and resources**. It is an activity that requires commitment, is a serious pursuit, and must be done as a primary goal of the team. Successful translation cannot be done as a "hobby." In order to support the software infrastructure, the team must respond to support tickets, organize training events and fix user-discovered bugs. As the community becomes international, these issues are amplified as time zone differences result in very short windows of high-bandwidth troubleshooting. This can cause support efforts for a single issue to stretch out from hours to days. Outreach and collaboration efforts were similarly constrained, making it more difficult to gain a foothold in a "new market" with established solutions, and build community. Each requires significant, timely work on behalf of the team; therefore, it must be valued appropriately. On a positive note, we believe the effort spent on operating production infrastructure is extremely valuable to the team; it assists the translation process by providing a mechanism for feedback from a wide community.

Translational Computing requires funding - but **often takes longer than any individual research project**. If we look at the "research heritage" of the HTCondor-CE, the HTCondor (previously, "Condor") system itself started as a cycle scavenging project in the 1980's [44] — over 30 years ago. Some of the most direct software contributions to the HTCondor-CE were the results from distinct research projects spanning a decade. The enabler is having a team interested in the translation and its outcomes — the PI-driven team is what allows the research products to be carried forward throughout the long-term translational work. Even then, this is not necessarily sufficient to do all the work — in the HTCondor-CE example, a core piece was the BLAHP which was a result of an external collaboration. We believe a focus on translation requires a cultural mindset from most CS research: while funding models may go in and out of vogue, the culture of the teams has a longer-term impact.

As long as there is a community to engage, the translational work will be ongoing. Almost no community — or its needs — is completely static. To be successful, TCS requires ongoing research throughout the lifetime of the project. For example, the OSG is currently going through a major overhaul of its authorization scheme to switch from GSI to a token-based infrastructure [45]. Since the beginning of the HTCondor-CE in 2012 and the first release in 2013, there has been ongoing collaboration with CHTC and others, allowing new research ideas to flow into the product. Feedback loops with the community provide valuable insight into future work; for the HTCondor-CE, the community helped redesign the configuration language used by the JobRouter, identify the need for a network configuration debugging tool, and helped guide the addition of Kubernetes-based packaging. This partnership was active through the years, resulting in 54 HTCSS tickets about the CE (bugs and improvements), 68 BLAHP tickets, and 227 HTCondor-CE tickets in the OSG. We look forward to sustaining this work for as long as there is a vibrant community around it.

5. Conclusions

The work by Livny and Melman reported in their 1982 paper [46] and recognized by the community [47] as the driver for research in adaptive load sharing policies was the starting point for the translational journey reported in this paper. The twists and turns, ups and downs, and disappointments and accomplishments of the journey took us from queuing theory and simulation models to a widely deployed Compute Entrypoint service are typical to the translational process. It takes a commitment that lasts years if not decades, spans organizations and continents, and sustains fluctuations in funding. We hope the experiences reported here will help CS researchers and domain science communities to engage in joint activities that translate advances in computing methodologies and technologies into better science. Our experience strongly supports our belief that such joint endeavors that are based on mutual trust and respect benefit the research of all parties; this trust helps mitigate the risk incurred by both sides due to engaging in a long-term translational process. Computer scientists benefit from demanding and committed users who evaluate new frameworks and technologies with real life applications in production environments and with research goals.

We are fully aware of the obstacles that translational projects face. They range from the culture of the Computer Science community to the funding models of the agencies and from the lack of TCS methodologies and frameworks to the difficulties in building and sustaining a team of software professionals in academia that provides stability and continuity. Most if not all of work of a translational project like the HTCondor-CE was done by staff members. Underpinning frameworks and technologies like the ClassAd language are the result of an earlier PhD work. One can view these obstacles as the missing infrastructure – intellectual and financial - that is needed to turn TCS into a mainstream academic practice. The value proposition of a TCS project is complex and multi-dimensional. It is likely to include both quantitative and qualitative metrics and arguments. We as computer scientists do poorly when it comes to qualitative argument. It is much easier to reason about expected latency or algorithm complexity than dependability or ease of use. This is especially true for translational projects – like the HTCondor-CE – that focus on mechanisms and not policies. We hope that the HTCondor-CE example for how translational work can be used to augment basic research with valuable experimental data, and the satisfaction of impact will mobilize the community to address these obstacles and to leverage the experience of similar translational projects. Training and building a TCS workforce and infrastructure will take a long-term commitment. Scientific discovery is waiting for us to take action.

Declaration of Competing Interest

This material is based upon work supported by the National Science Foundation under Grant No. 1321762 and 1148698. Beyond this support, the authors of this manuscript have no further conflicts of interest to report.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1321762 and 1148698.

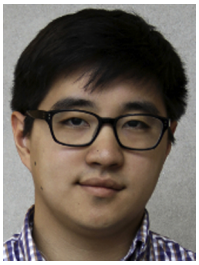
References

[1] B. Lin, B. Bockelman, et al., HTCondor-CE v4.3.0-2, May, 2020, https://doi.org/10.5281/zenodo.3862643, https://github.com/htcondor/htcondor-ce.
[2] D. Abramson, M. Parashar, Translational research in computer science, Computer 52 (September (9)) (2019) 16–23, https://doi.org/10.1109/MC.2019.2925650.
[3] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the condor experience, Concurr. Comput. Pract. Exp. 17 (February-April (2-4)) (2005) 323–356, https://doi.org/10.1002/cpe.938.
[4] M.J. Litzkow, M. Livny, M.W. Mutka, Condor-a hunter of idle workstations, 1988] Proceedings. The 8th International Conference on Distributed (1988) 104–111, https://doi.org/10.1109/DCS.1988.12507.
[5] R. Raman, M. Livny, M. Solomon, Matchmaking: distributed resource management for high throughput computing, Proceedings. The Seventh International Symposium on High Performance Distributed Computing (Cat. No.98TB100244) (1998) 140–146, https://doi.org/10.1109/HPDC.1998.709966.
[6] L. Evans, P. Bryant, LHC machine, JINST 3 (2008), S08001, https://doi.org/10.1088/1748-0221/3/08/S08001.
[7] The WLCG Collaboration, https://wlcg-public.web.cern.ch/.
[8] J. White, gLite and condor present and future, Presentation at HTCondor Week (2006). https://research.cs.wisc.edu/htcondor/CondorWeek2006/presentations/white_egee.pdf.
[9] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimäki, O. Smirnova, A. Wäänänen, Advanced resource connector middleware for lightweight computational grids, Future Gener. Comput. Syst. 23 (2) (2007) 219–240, https://doi.org/10.1016/j.future.2006.05.008. http://www.sciencedirect.com/science/article/pii/S0167739X06001178.

[10] D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, J. Pruyne, A worldwide flock of Condors: load sharing among workstation clusters, Future Gener. Comput. Syst. 12 (1996) 53–65, https://doi.org/10.1016/s0167-8191(98)00079-9.
[11] J. Basney, M. Livny, Paolo Mazzanti, Utilizing widely distributed computational resources efficiently with execution domains, Comput. Phys. Commun. 140 (October (1-2)) (2001) 246.
[12] R. Gardner, grid3 : an application grid laboratory for science, Proceedings Computing in High Energy Physics and Nuclear Physics (2004) 18, https://doi.org/10.5170/CERN-2005-002.18.
[13] The LHC experiments Committee, Technical Proposal for CMS Computing, Tech. Rep. CERN-LHCC-96-045, CERN, Geneva, 1996, https://cds.cern.ch/record/322321.
[14] M. Aderholz, K. Amako, E. Augé, G. Bagliesi, L. Barone, G. Battistoni, M. Bernardi, M. Boschini, A. Brunengo, J.J. Bunn, J. Butler, M. Campanella, P. Capiluppi, F. Carminati, M. D'Amato, M. Dameri, A. Di Mattia, A.E. Dorokhov, G. Erbacci, U. Gasparini, F. Gagliardi, I. Gaines, P. Gálvez, A. Ghiselli, J. Gordon, C. Grandi, F. Harris, K. Holtman, V. Karimäki, Y. Karita, J.T. Klem, I. Legrand, M. Leltchouk, D. Linglin, P. Lubrano, L. Luminari, A.L. Maslennikov, A. Mattasoglio, M. Michelotto, I.C. McArthur, Y. Morita, A. Nazarenko, H. Newman, V. O'Dell, S. W. O'Neale, B. Osculati, M. Pepé, L. Perini, J.L. Pinfold, R. Pordes, F. Prelz, A. Putzer, S. Resconi, L. Robertson, S. Rolli, T. Sasaki, H. Sato, L. Servoli, R. D. Schaffer, T.L. Schalk, M. Sgaravatto, J. Shiers, L. Silvestris, G.P. Siroli, K. Sliwa, T. Smith, R. Somigliana, C. Stanescu, H.E. Stockinger, D. Ugolotti, E. Valente, C. Vistoli, I.M. Willers, R.P. Wilkinson, D.O. Williams, Models of Networked Analysis at Regional Centres for LHC Experiments (MONARC), Phase 2 Report, 24th March 2000, Tech. Rep. CERN-LCB-2000-001. KEK-2000-8, CERN, Geneva, 2000, https://cds.cern.ch/record/510694.
[15] K. Bos, N. Brook, D. Duellmann, C. Eck, I. Fisk, D. Foster, B. Gibbard, C. Grandi, F. Grey, J. Harvey, A. Heiss, F. Hemmer, S. Jarp, R. Jones, D. Kelsey, J. Knobloch, M. Lamanna, H. Marten, P. Mato Vila, F. Ould-Saada, B. Panzer-Steindel, L. Perini, L. Robertson, Y. Schutz, U. Schwickerath, J. Shiers, T. Wenaus, LHC Computing Grid: Technical Design Report, Version 1.06 (20 Jun 2005), Technical Design Report LCG, CERN, Geneva, 2005, http://cds.cern.ch/record/840543.
[16] G. Bayatyan, M.D. Negra, A. Foà, Hervé, A. Petrilli, CMS computing: Technical Design Report, submitted on 31 May 2005, CERN, Geneva, 2005, https://cds.cern.ch/record/838359.
[17] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1999. ISBN 978-1-55860-475-9.
[18] I. Foster, C. Kesselman, The Globus project: a status report, Proceedings Seventh Heterogeneous Computing Workshop (HCW'98) (1998) 4–18, https://doi.org/10.1109/HCW.1998.666541.
[19] I. Sfiligoi, glideinWMS—a generic pilot-based workload management system, J. Phys. Conf. Ser. 119 (6) (2008), 062044,, https://doi.org/10.1088/1742-6596/119/6/062044.
[20] I. Sfiligoi, D.C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, F. Wuerthwein, The pilot way to grid resources using glideinWMS, in: 2009 WRI World Congress on Computer Science and Information Engineering, 2, 2009, pp. 428–432, https://doi.org/10.1109/CSIE.2009.950.
[21] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-G: a computation management agent for multi-institutional grids, Proceedings 10th IEEE International Symposium on High Performance Distributed Computing (2001) 55–63, https://doi.org/10.1109/HPDC.2001.945176.
[22] M. Mezzadri, F. Prelz, D. Rebatto, Job submission and control on a generic batch system: the BLAH experience, J. Phys. Conf. Ser. 331 (6) (2011), 062039,, https://doi.org/10.1088/1742-6596/331/6/062039.
[23] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buni, F. Hemmer, A. Di Meglio, Edlund, P. Buncic, Programming the Grid with gLite*, Comput. Methods Sci. Technol. 12 (1) (2006) 33–45, https://doi.org/10.12921/cmst.2006.12.01.33-45.
[24] D. Bradley, S. Dasu, M. Livny, A. Mohapatra, T. Tannenbaum, G. Thain, Condor enhancements for a rapid-response adaptive computing environment for LHC, J. Phys. Conf. Ser. 219 (2010), https://doi.org/10.1088/1742-6596/219/6/062035.
[25] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke, Security for Grid services, High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on (2003) 48–57, https://doi.org/10.1109/HPDC.2003.1210015.
[26] A. Withers, B. Bockelman, D. Weitzel, D. Brown, J. Gaynor, J. Basney, T. Tannenbaum, Z. Miller, SciTokens: capability-based secure access to remote scientific data, Proceedings of the Practice and Experience on Advanced Research Computing (2018) 1–8, https://doi.org/10.1145/3219104.3219135.
[27] C.T. Gilliland, J. White, B. Gee, R. Kreeftmeijer-Vegter, F. Bietrix, A.E. Ussi, M. Hajduch, P. Kocis, N. Chiba, R. Hirasawa, M. Suematsu, J. Bryans, S. Newman, M.D. Hall, C.P. Austin, The fundamental characteristics of a translational scientist, ACS Pharmacol. Transl. Sci. 2 (3) (2019) 213–216, https://doi.org/10.1021/acsptsci.9b00022.
[28] J. White, gLite and Condor Present and Future, Condor Week, 2006, https://research.cs.wisc.edu/htcondor/CondorWeek2006/presentations/whiteegee.pdf.
[29] L. Field, M. Schulz, Grid Deployment Experiences: the Path to a Production Quality LDAP Based Grid Information System, 2005, https://doi.org/10.5170/CERN-2005-002.723. http://cds.cern.ch/record/865688.
[30] P. Andreetto, S. Bertocco, F. Capannini, M. Cecchi, A. Dorigo, E. Frizziero, A. Gianelle, F. Giacomini, M. Mezzadri, S. Monforte, F. Prelz, E. Molinari, D. Rebatto, M. Sgaravatto, L. Zangrando, Status and developments of the CREAM computing element service, J. Phys. Conf. Ser. 331 (2011), 062024,, https://doi.org/10.1088/17426596/331/6/062024.
[31] M. Feller, I. Foster, S. Martin, GT4 GRAM: a functionality and performance study, Proceedings of TeraGrid Conference (2007).
[32] I. Sfiligoi, S. Padhi, Evaluation of New Compute Element Software for the Open Science Grid: GRAM5 and CREAM, OSG Document 1006,, 2010, https://osg-docdb.opensciencegrid.org/0010/001006/001/gram5_cream_1004.pdf.
[33] D. Groep, O. Koeroo, G. Venekamp, Grid Site Access Control and Credential Mapping to the Unix Domain, Nikhef PDP Tech. Rep. https://www.nikhef.nl/grid/LCMAPS.
[34] A. Yoo, M. Jette, M. Grondona, Job scheduling strategies for parallel processing. Volume 2862 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 44–60.
[35] B. Bockelman, T. Cartwright, J. Frey, E. Fajardo, B. Lin, M. Selmeci, T. Tannenbaum, M. Zvada, Commissioning the HTCondor-CE for the open science grid, J. Phys. Conf. Ser. 664 (2020), 062003,, https://doi.org/10.1088/1742-6596/664/6/062003.
[36] Byrom, Rob & Cordenonsib, Roney & Cornwall, Linda & Craig, Martin & Djaoui, Abdeslem & Duncan, Alastair & Fisher, Stephen & Gordon, John & Hicks, Steve & Kant, Dave & Leakec, Jason & Middleton, Robin, APEL: An implementation of grid accounting using R-GMA, 2005.
[37] V. Tschopp, Argus: the EMI authorization service, Presentation in EGI User Forum (2011). https://twiki.cern.ch/twiki/pub/EGEE/AuthorizationFrameworkArchive/20110412-EGI_UF_2011-Argus.ppt.
[38] A. Adams, K. Avila, J. Basney, D. Brunson, R. Cowles, J. Dopheide, T. Fleury, E. Heymann, F. Hudson, C. Jackson, R. Kiser, M. Krenz, J. Marsteller, B.P. Miller, S. Piesert, S. Russell, S. Sons, V. Welch, J. Zage, Trusted CI experiences in cybersecurity and service to open science, PEARC'19: Practice and Experience in Advanced Research Computing (2019), https://doi.org/10.1145/3332186.3340601.
[39] B. Lin, HTCondor-CE: managing the grid with HTCondor, Presentation at HEPiX Fall 2015 Workshop (2015). https://indico.cern.ch/event/384358/contributions/909195/.
[40] B. Bockelman, HTCondor-CE overview and architecture, Presentation at Workshop for HTCondor and ARC-CE Users (2016). https://indico.cern.ch/event/467075/contributions/1143794/.
[41] B. Bockelman, Progress report on the HTCondor-CE, Presentation at European HTCondor Workshop 2017 (2017). https://indico.cern.ch/event/611296/contributions/2608194/.
[42] J. Frey, HTCondor-CE overview and architecture, Presentation at European HTCondor Workshop 2018 (2018). https://indico.cern.ch/event/733513/contributions/3117188/.
[43] G. Thain, HTCondor-CE overview: from clusters to grids, Presentation at European HTCondor Workshop 2019 (2019). https://indico.cern.ch/event/817927/.
[44] HTCondor & ARC Workshop. http://event.twgrid.org/isgc2019/index.html.
[45] A. Withers, B. Bockelman, D. Weitzel, D.A. Brown, J. Gaynor, J. Basney, T. Tannenbaum, Z. Miller, SciTokens: capability-based secure access to remote scientific data, in: PEARC '18: Practice and Experience in Advanced Research Computing, Pittsburgh, PA, USA, July, 2018, https://doi.org/10.1145/3219104.3219135.
[46] M. Livny, M. Melman, Load balancing in homogeneous broadcast distributed systems. ACM SIGMETRICS Performance Evaluation Review, 1982, https://doi.org/10.1145/1010631.801689. April.
[47] D.L. Eager, E.D. Lazowska, J. Zahorjan, A comparison of receiver-initiated and sender-initiated adaptive load sharing, Acm Sigmetrics Perform. Eval. Rev. (August) (1985), https://doi.org/10.1145/317786.

Brian Bockelman received a B.S. in Mathematics in 2003 from the University of West Georgia and M.S. and pH.D. degrees in Mathematics and Mathematics & Computer Science from the University of Nebraska-Lincoln (UNL) in 2005 and 2008, respectively. He has served as research faculty at UNL and, since 2019, as an associate scientist at the Morgridge Institute for Research. Dr. Bockelman's research focuses on distributed computing and data management ranging from large-scale scientific endeavors such as the CMS and LIGO experiments to enabling PI-driven groups on the Open Science Grid.

Miron Livny received a B.Sc. degree in Physics and Mathematics in 1975 from the Hebrew University and M.Sc. and pH. D. degrees in Computer Science from the Weizmann Institute of Science in 1978 and 1984, respectively. Since 1983 he has been on the Computer Sciences Department faculty at the University of Wisconsin-Madison, where he is currently the John P. Morgridge Professor of Computer Science, the director of the Center for High Throughput Computing (CHTC), is leading the HTCondor project and serves as the technical director of the Open Science Grid (OSG). He is a member of the scientific leadership team of the Morgridge Institute of Research and is serving as the Chief Technology Officer of the Wisconsin Institutes of Discovery. Dr. Livny's research focuses on distributed processing and data management systems and involves close collaboration with researchers from a wide spectrum of disciplines. He pioneered the area of High Throughput Computing (HTC) and developed frameworks and software tools that have been widely adopted by academic and commercial organizations around the world. Livny is the recipient of the 2006 ACM SIGMOD Test of Time Award the 2013 HPDC Achievement Award and the 2020 IEEE TCDP Outstanding Technical Achievement Award.



Brian Lin received a B.Sc. in Atmospheric Sciences and Physics from McGill University and is the current Software Area Coordinator for the Open Science Grid (OSG). He leads the team responsible for packaging, integrating, and supporting the OSG software stack. Lin is the current maintainer of the HTCondor-CE, having overseen its development in the last five years.

Francesco Prelz works for the Italian National Institute for Nuclear Physics (INFN) in Milan as a director of technology. He has a long history of work in distributed computing for the Large Hadron Collider (LHC), contributing to European projects from the European Data Grid to the gLite project to the European Middleware Initiative. He is the driving force behind the BLAHP, which serves to integrate middleware such as the CREAM CE or the HTCondor-CE and local batch systems.