CHEP 2019

Exploiting network restricted compute resources with HTCondor: a CMS experiment experience

J. Flix

C. Acosta-Silva, A. Delgado Peris, J. Frey, J. M. Hernández, M. Livny, A. Pérez-Calero Yzquierdo, T. Tannenbaum

PIC
port d'informació científica

Institut de Física d'Altes Energies
IFAE
Ciemat
Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

Introduction

PIC
port d'informació
científica

Large CPU resources in **supercomputing facilities (HPCs)** and much more to come

- Roadmap towards ExaFlop machines worldwide
- HPCs could help finding a solution for the HL-LHC CPU problem

Using HPC for LHC High Throughput Computing (HTC) is **challenging** with **non-negligible** integration work

Simulation is the workflow (CPU-bound) that best adapts to HPC (**significant use of resources**)

CMS has integrated some HPCs for exploitation, but small CPU fraction is utilized overall

Ongoing work to profit from the sizeable CPU resources offered by the Spanish supercomputing network (**RES**), where the Barcelona Supercomputing Center (**BSC**) provides most of the resources

BSC overview

PIC
port d'informació científica

The **Barcelona Supercomputing Center** (www.bsc.es) is the main HPC center in Spain

Their biggest general-purpose cluster is called **MareNostrum(4)**:

- **CPU:** 48 racks with 3,456 nodes, each with two Intel Xeon Platinum chips, each with 24 processors, amounting to a total of 165,888 processors
- **Memory:** 390 TB
- **Disk:** 24 PB
- MN4 peak power is **11.15 Petaflops**

They use **Slurm** as batch system and nodes **OS is SUSE Linux Enterprise** Server 12 SP2

Additional **specialised clusters** (IBM POWER9 processors and NVIDIA Volta GPUs, Intel Knights Hill (KNH) processors, 64 bit ARMv8 processors)

More at https://www.bsc.es/marenostrum/marenostrum/technical-information

BSC overview

BSC will soon be the proud owners of one of the Europe's first **pre-exascale supercomputers**

The new supercomputer is expected to deliver **200 peak Petaflops***

* The system will no longer fit in the Chapel. Some of racks will be placed in the lower floors of BSC's new building, a few meters away

BSC constraints to run WLCG jobs

PIC
port d'informació científica

‘Call for resource allocation’ available, but not suitable to get a guaranteed share of resources for a long-term resource exploitation

LHC applications are NOT really well-suited for HPC

- Applications are not optimised to run on their CPU architectures and OS
- No large parallelization (no use of fast node interconnects)
- No essential use of accelerators (GPU, FPGA)

BSC has **typical HPC limitations** to run WLCG jobs:

- BSC execute nodes have no external connectivity
- The login machine(s) allow incoming connection through the ssh port (5’ cpu limit per process)
- No storage element as used in WLCG standards → there is a shared disk (GPFS) mounted on execute nodes and login machines, and externally available via sshfs

Substantial **integration work** to make HPC work for HTC

ATLAS antecedents on BSC

PIC
port d'informació científica

ATLAS has achieved job submission to BSC via custom **ARC-CE at PIC**, acting as a bridge between Panda and local BSC's Slurm (since mid-2018; in 2019 PIC granted with 2.75M hours)

- Slurm commands are executed remotely using ssh

The input/output files are copied from/to using an **sshfs volume mounted at PIC** for the MareNostrum disk user space (GPFS)

The **ARC-CE moves data** to PIC storage (ARC cache) and registers the datasets on ATLAS Rucio

Jobs are run inside a **fat (500 MB) singularity image**, which provides CentOS 7 OS, the ATLAS software versions, and conditions data (static)

Each submitted **job allocates a full node** (48 cores) for (scheduling) efficiency

Jobs are submitted with a predefined payload (**no late binding**)

Connectivity problem for CMS

PIC
port d'informació científica

CMS submit **pilot jobs** to sites. These interact with the **CMS Global Pool** to:

- Fetch configuration and validation scripts to create and run the HTCondor startd for CMS
- Connect to the collector/negotiator to get payloads to execute (late binding)

Typically jobs require:

- Compute nodes need (a few) ports opened
- CVMFS is needed for CMSSW versions and Singularity images
- Payloads get conditions from Squid Caches (port 80)
- Simulation DIGI workflow typically needs to read multiple files via XRootD (to add pile-up)

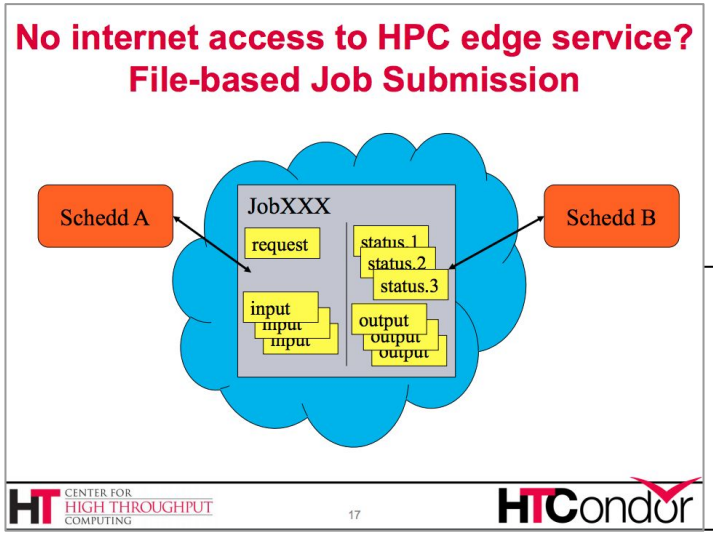
BSC context:

- Execute nodes do not have open ports, hence pilots cannot get payloads from Global Pool
- Not possible to install edge services (Squids, CVMFS, ...)
- There is a shared GPFS area mounted on all of the BSC compute nodes and in the UI where users have access

Jumping the network barrier

It's clear that most (all?) of the HPC-exploiting experiences in CMS have required **integration efforts** as well as tuning and some **flexibility from HPC providers**

In order to succeed in the BSC very difficult scenario we can make use of a HTCondor developers idea: **use a shared FS as control path for HTCondor**



T. Tannenbaum, introduced concept:
"What's new in HTCondor", HTCondor Week 2018
https://agenda.hep.wisc.edu/event/1201/session/8/
T. Tannenbaum, deployment:
"What's new in HTCondor", EU HTCondor Workshop 2019
https://indico.cern.ch/event/817927/contributions/3570445/

PIC - HTCondor devs collaboration

A collaboration was formalized during the **September'18 RAL HTCondor workshop**
[Miron Livny, Todd Tannenbaum, Jaime Frey, Antonio Pérez-Calero, Carles Acosta, José Flix]

Goal: use "HTCondor via FS" idea to solve the BSC case, but with the aim to develop a general solution for similar difficult scenarios elsewhere!

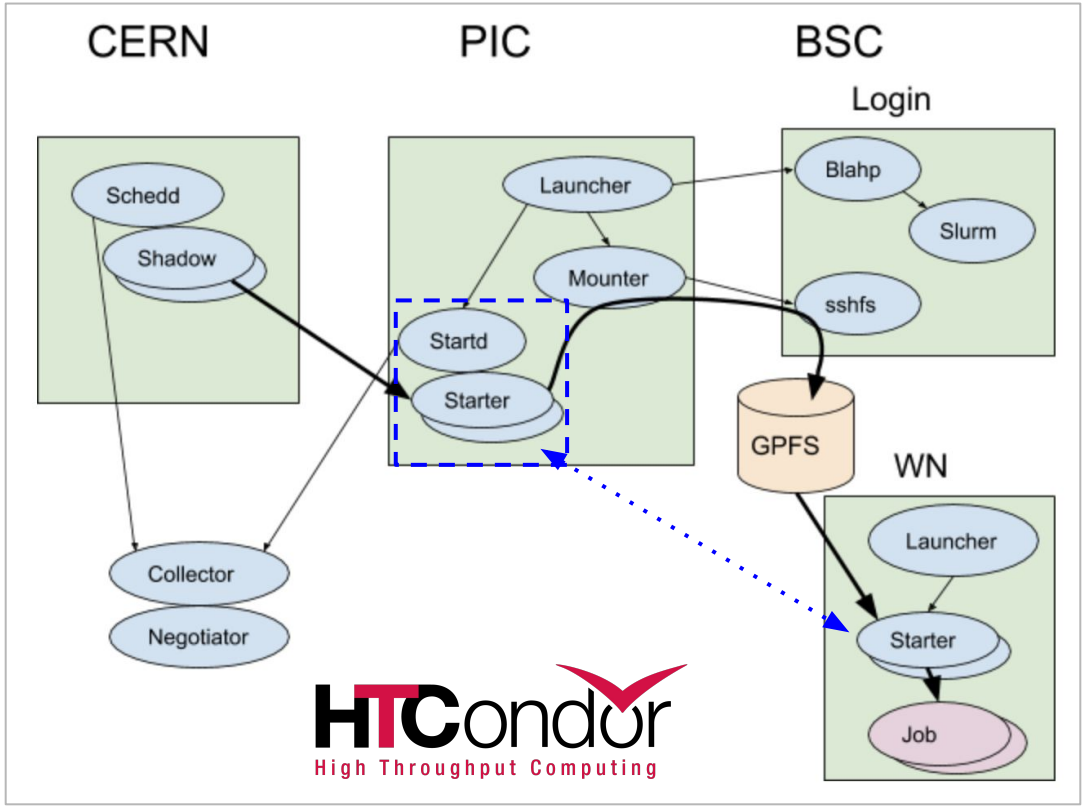
Progress: regular meetings since then, a number of documents addressing each piece of the puzzle, and a testbed deployed in PIC to test the new features, interacting with BSC, and connected to the CMS Global Pool

A model for HTCondor via shared-FS

Main idea: a bridge node at PIC which allows access to starter processes running in the BSC nodes, mirroring starter at PIC

The startd process remains at PIC, where it can be accessed to negotiate and **keep late binding**

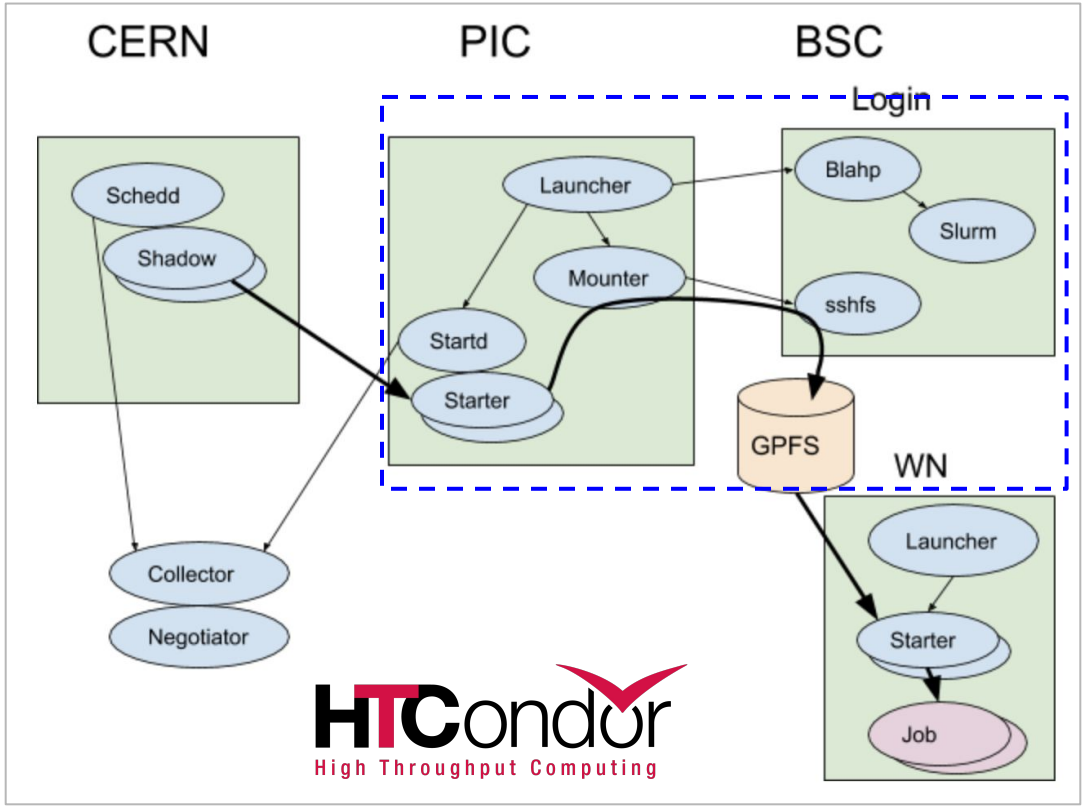
From a functional perspective, the **node at BSC joins the HTCondor pool at PIC**



The model at work

The **bridge node** interacts with BSC login node:

- **submit jobs to BSC** Slurm batch system using blahp over ssh (a setup known as **Bosco**)
- Also handles file transfers over ssh to BSC GPFS (e.g. CMS payloads as input sandboxes)



The model at work

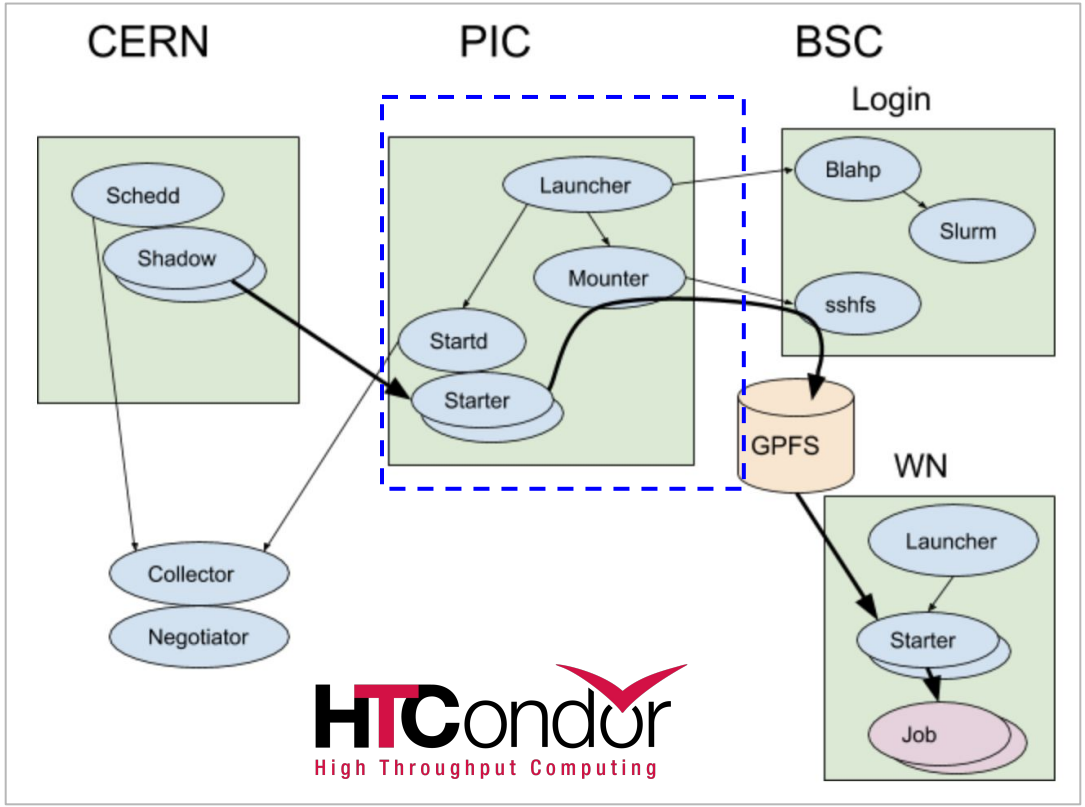
For each node running the Slurm job, a **condor_startd** daemon is spawned on the PIC server

Startds are configured to act like they have the **hardware resources** (cpus, memory, disk) of a single BSC execute node

Each startd sets up an **sshfs mount** on the PIC server of BCS's GPFS file system (via BSC login node), creating a series of **rendezvous directories** in the FS, one per BSC node

Startd's join the pool collector but advertise themselves as being unavailable for matchmaking

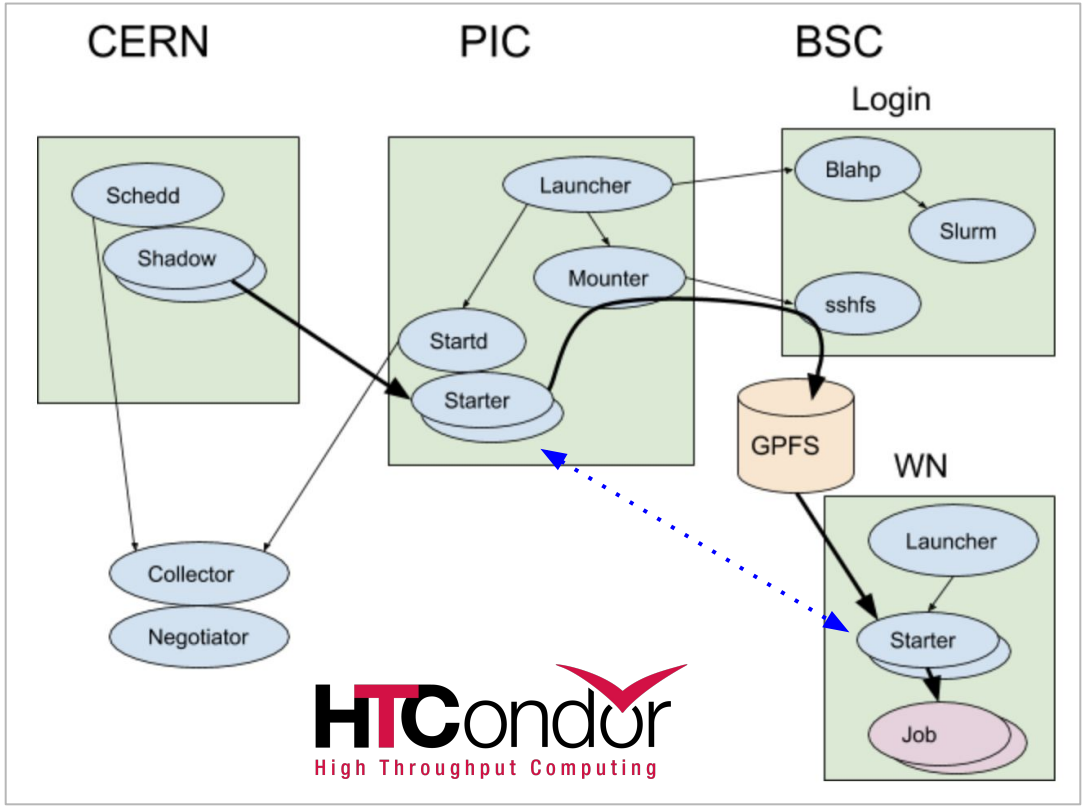
Once the Slurm job starts running, startds re-advertise as r**eady to accept jobs**



The model at work

Once Slurm jobs start running the usual HTCondor machinery progresses:

- Matchmaking done at the **local negotiator**
- CMS submit machines (schedds) connect to the startds via **flocking**
- **starters** are spawned at PIC and payload sandboxes are transferred from the schedd
- Then, instead of spawning the payload job directly, the starter copies the **job_ad and input sandbox** to the corresponding rendezvous directory of its startd in GPFS
- PIC starters wait for the **output sandbox** and a job_ad summarizing the execution to appear in the rendezvous directory
- These **files are copied out** of FS and normal post-job-execution steps proceed (transfer the output sandbox to the submit machine, etc...)



Current Status and Outlook

PIC
port d'informació
científica

PIC setup deployed, up and running. The **HTCondor bridge** has been built and tested (jobs to slurm are sent manually, though), and it has been integrated into the a test CMS Global Pool - blind test jobs have been run to validate the new functionalities

Recipes to include soon:

- Create CMS Fat Singularity images with all relevant CMSSW versions in CentOS7 - pre-place it in BSC GPFS
- Get conditions from squid servers to a sqlite file - modify payloads to read this file from BSC GPFS
- How to inject particular CMS jobs to BSC, via HTCondor ClassAdds (MC GEN-SIM, which does not require input files)

Next*:

- Automate the slurm job submissions from PIC HTCondor bridge, in order to get resources
- When to submit job requests to BSC's Slurm Matchmaking in PIC negotiator based on payloads being correctly tagged
- Moving payload job output data at BSC to HTCondor bridge and transfer it to PIC SE (pos-process)

* Expecting to have a functional system by the end of 2019

Long term collaboration with BSC

Meanwhile, a **collaboration agreement with BSC** is under preparation, since they have been involved in many of the discussions:

- To be **approved soon** by BSC management board
- LHC computing would be a **strategic project** with a guaranteed share of BSC resources
- Aim to provide in the **long term** the CPU required to Spain for **LHC data simulation**
- This will require **intense integration program** to use BSC resources efficiently (networking, security, hardware, etc)

Quite some work ahead towards transparent exploitation of BSC resources for CMS!

PIC
port d'informació
científica

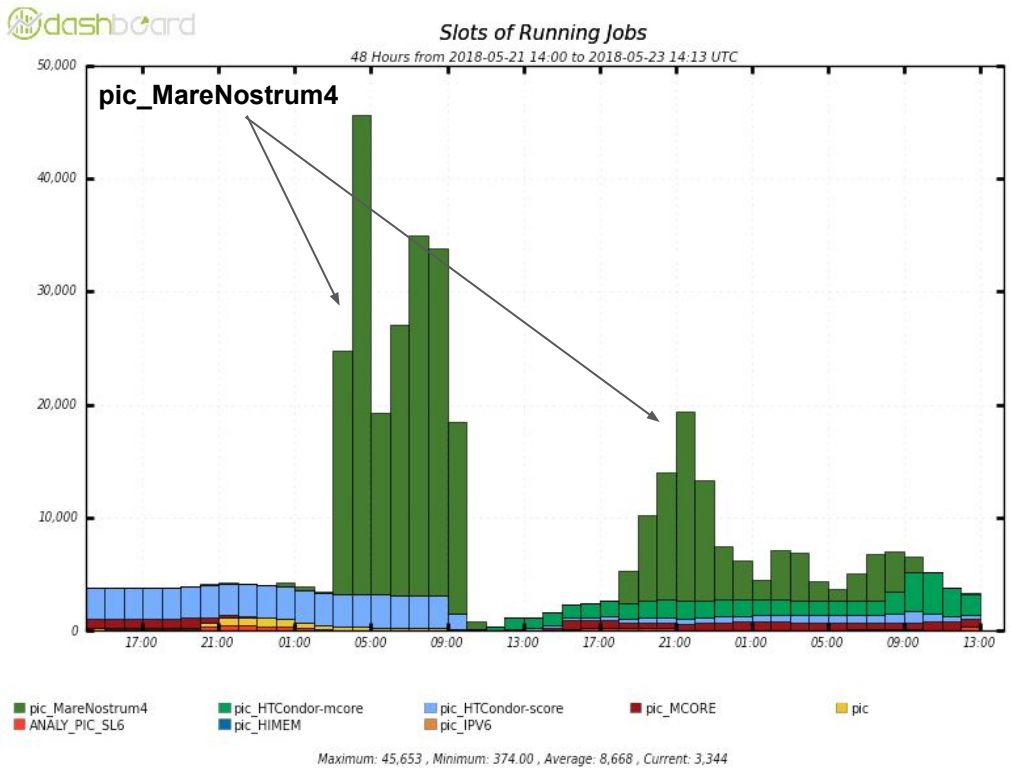
Thanks!
Questions?

BACKUPS

Exploiting network restricted compute resources with HTCondor: a CMS experiment experience

Abstract: In view of the increasing computing needs for the HL-LHC era, the LHC experiments are exploring new ways to access, integrate and use non-Grid compute resources. Accessing and making efficient use of Cloud and supercomputer (HPC) resources present a diversity of challenges. In particular, network limitations from the compute nodes in HPC centers impede CMS experiment pilot jobs to connect to its central HTCondor pool to receive the actual payload jobs to be executed. To cope with this limitation, new features have been developed in both HTCondor and the CMS resource scheduling and workload management infrastructure. In this novel approach, a bridge is set up outside the HPC center and the communications between HTCondor daemons are relayed through a shared file system. We have used this strategy to exploit the Barcelona Supercomputing Center (BSC) resources, the main Spanish HPC site. CMS payloads are claimed by HTCondor startd daemons running at the nearby PIC Tier-1 center and routed to BSC compute nodes through the bridge. This fully enables the connectivity of CMS HTCondor-based central infrastructure to BSC resources via PIC HTCondor pool. Other challenges have included building custom Singularity images with CMS software releases, bringing conditions data to payload jobs, and custom data handling between BSC and PIC. This contribution describes this technical prototype, its deployment, the functionality and scalability tests performed, along with the results obtained when exploiting the BSC resources using these novel approaches. A key aspect of the technique described in this contribution is that it could be universally employed in similar HPC environments elsewhere.

Integration of HPC resources [ATLAS]



Tests on the **MareNostrum HPC** integration in the ATLAS production system started in April 2018 in joint collaboration with IFIC Tier2 site

Since then, we have received hours to exploit Spanish HPC's (**RES** and **PRACE**):

In 2019, PIC has been granted **2.75 million hours in the MareNostrum 4 HPC**

Two types of payload submission:

- 1 job = 1 full node (48 cores)
- 1 job = 50 nodes using MPI/Yoda (2400 cores)

Data async. transferred to PIC and registered into ATLAS Rucio system

Tested transfer mode using **globus-url-copy** with ssh as authentication (no certificates) which is standard for HPC sites

Integration of HPC resources [CMS]

For CMS, we are **working in a model**, similar to ATLAS, in close collaboration with HTCondor developers

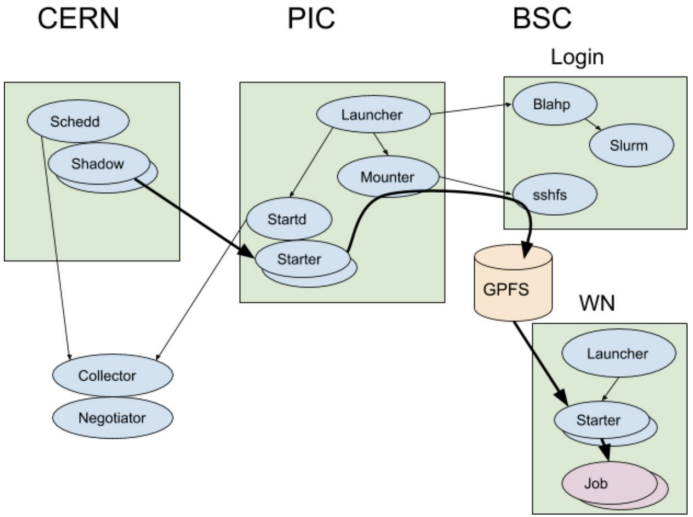
Lack of outbound network connectivity in nodes reduces flexibility for CMS… since CMS pilot jobs connect to global CMS HTCondor pool to get the actual payloads

→ Developed and tested a mechanism to interconnect HTCondor pools through **shared file systems**

→ Dedicated **PIC testbed** in place, some tests run connected to CMS Global Pool

→ We need to **instrument** the CMS **payloads** so they can run at BSC (sqlite file for conditions, singularity image, data export handling)

→ Goal is to incorporate BSC resources and run CMS simulations



HTCondor High Throughput Computing

CHEP 2019

What's new in HTCondor? [EU HTCondor Workshop 2019]

CMS jobs network needs:

Pilot needs:

- CMS submit pilot jobs to sites. These interact with the CMS Global Pool to:
  - Fetch configuration and validation scripts to create and run the HTCondor startd for CMS
  - Payload distribution: Connect to the collector/negotiator to get payloads to execute (late binding)

Payload job needs:

- Software distribution: CVMFS is needed for CMSSW versions and Singularity images
- Database access: Payloads get conditions from Squid Caches (port 80)
- Data input and output:
  - Data processing tasks: input data streamed to payload
  - Simulation tasks: still need to read multiple input data files, streamed via XRootD (to add pile-up)
  - Output insertion on CMS data catalogues
- Communication with central services (monitoring, log collect, etc)

Alternatives in case of limited connectivity

- Slot validation: “emulated pilot” mechanism to download CMS set of validation scripts in order to prepare the slot to be used by CMS jobs
  - Basically, download validation scripts from the glideinWMS factory
- Software: container images made locally available, including
  - Operating system plus dependencies
  - Pre-packaged, payload-dependent, CMS SW images
  - Install software in shared FS?
- Database: Extract conditions data and make it available locally as additional input files
- Input data: either
  - Limit the needs for input data by specifically choosing tasks that do not require it (pure simulation)
    - Limits flexibility and usability of these resources
  - Data pre-placement to local storage: dedicated agent to feed input files and get output files