Exploitation of network-restricted resources at the Barcelona Supercomputer Center by CMS

J. Flix

C. Acosta-Silva, A. Delgado Peris, J. M. Hernández, A. Pérez-Calero Yzquierdo, E. Pineda, I. Villalonga

PIC
port d'informació
científica

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

Motivation and context

PIC
port d'informació científica

HPC facilities are attractive for **WLCG** computing

- To help cover increasing CPU needs despite flat funding
- Lot of public funding went to HPCs
- Several ongoing HPC integration efforts in CMS

BSC - Barcelona Supercomputing Center

- Largest HPC center in Spain
- Current MareNostrum 4 (*MN4*) general-purpose cluster:
  - 11.5 Petaflops (165,888 CPUs), 390 TB RAM, 24 PB disk
  - SLURM as batch system, SUSE Linux Enterprise as OS
  - GPFS as storage back-end (mounted on login/compute nodes)
- Next MareNostrum5 (~17xMN4, ~200 petaflops), available in 2023 (?)
  - One of Europe's first pre-exascale supercomputers

Motivation and context

In 2020 BSC designated LHC computing as a **strategic project**

- Agreement promoted by WLCG-ES community and funding agency

Allocations of up to a **7% share of MN4 for LHC**

- ~70M coreHours/year
- ~20M hours allocation for CMS in 2022

Potentially, very **significant contribution** for LHC computing in Spain

- Comparable e.g to all CMS simulation needs in the country

The challenges for CMS

PIC
port d'informació científica

BSC imposes very **restrictive network connectivity** conditions

- No incoming *or outgoing* connectivity from compute nodes
- Only incoming SSH/SSHFS communication through login nodes
- A shared disk (GPFS) mounted on execute nodes and login machines - accessible from outside via sshfs
- No services can be deployed on edge/privileged nodes

This is a **major obstacle for CMS** workloads

- Pilot with late binding model execution of payloads
  - Workload management system (glideinWMS - HTCondor services)
- Access to external services
  - Application software (CVMFS) & Conditions data (FrontierDB)
- Consuming and producing experiment data
  - Input/output data files (Storage Elements)

Connecting BSC to CMS Global Pool

HTCondor major development: modify CMS resource provisioning, job scheduling and execution framework (HTCondor) to use a **shared file system as communication layer**

- Split-starter model, presented at **CHEP19**
- Requires a bridge service at PIC to connect CMS WMS and BSC

A collaboration was formalized during the September'18 RAL HTCondor workshop

[Miron Livny, Todd Tannenbaum, Jaime Frey, Antonio Pérez-Calero, Carles Acosta, José Flix]

EPJ Web of Conferences 245, 09007 (2020)
https://doi.org/10.1051/epjconf/202024509007

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

Carles Acosta-Silva[3], Antonio Delgado Peris[2], José Flix Molina[1,2*], Jaime Frey[4], José M. Hernández[2], Miron Livny[4], Antonio Pérez-Calero Yzquierdo[1,2] and Todd Tannenbaum[4]

[1] Port d'Informació Científica (PIC), Barcelona, Spain
[2] Centro de Investigaciones Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[3] Institut de Física d'Altes Energies (IFAE), Barcelona, Spain
[4] University of Wisconsin-Madison, Madison WI, USA

[*] e-mail: jose.flix.molina@cern.ch

Published online: 16 November 2020

The HTCondor split-starter at work

Use communication via FS to replace WAN

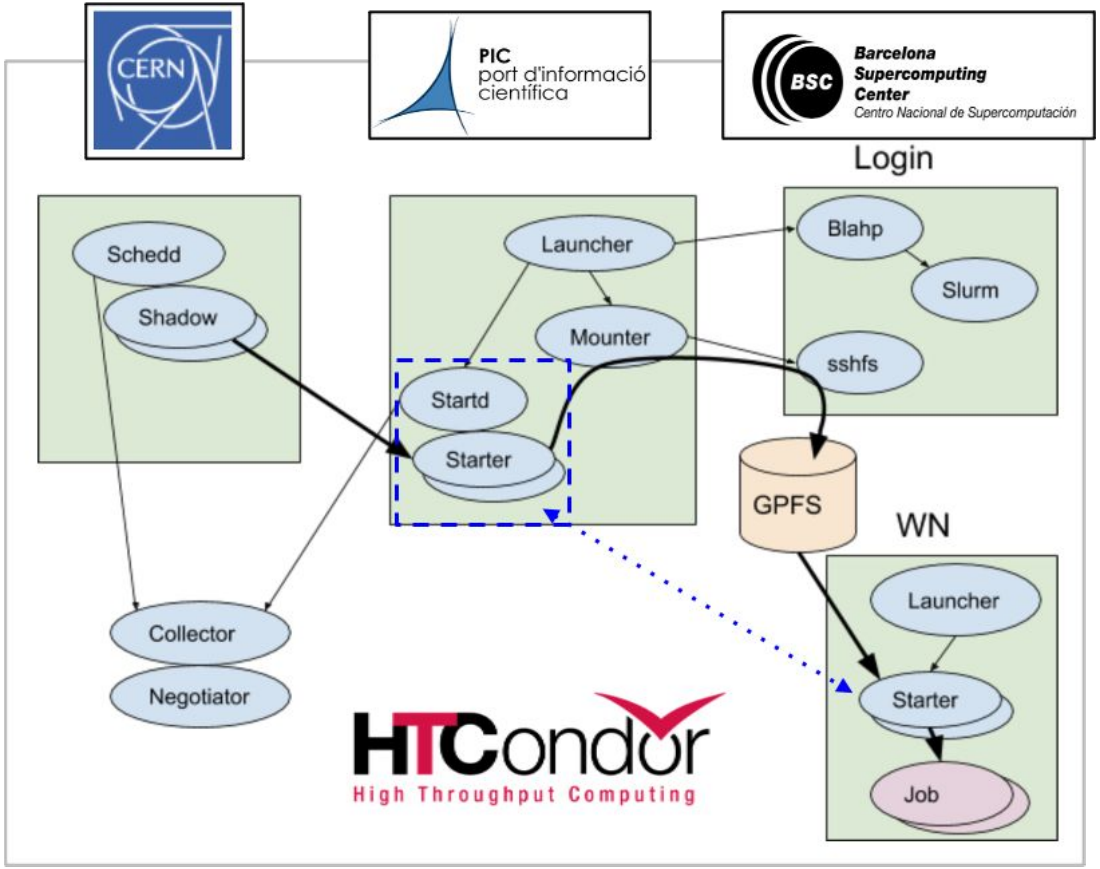
A **bridge node** installed at **PIC** submits **Slurm jobs to BSC** that instantiate **HTCondor starters** on the BSC nodes, which connect back to **startd processes** running at PIC

Slurm job at BSC behaves as a "**glidein**"

The **startd** process at PIC can join a local condor pool, or connect ot the CMS Global Pool, enabling **matchmaking** by **CMS job schedds at CERN**

Job input sandbox, status, etc, **passed as .tar files from bridge to WN via gpfs**

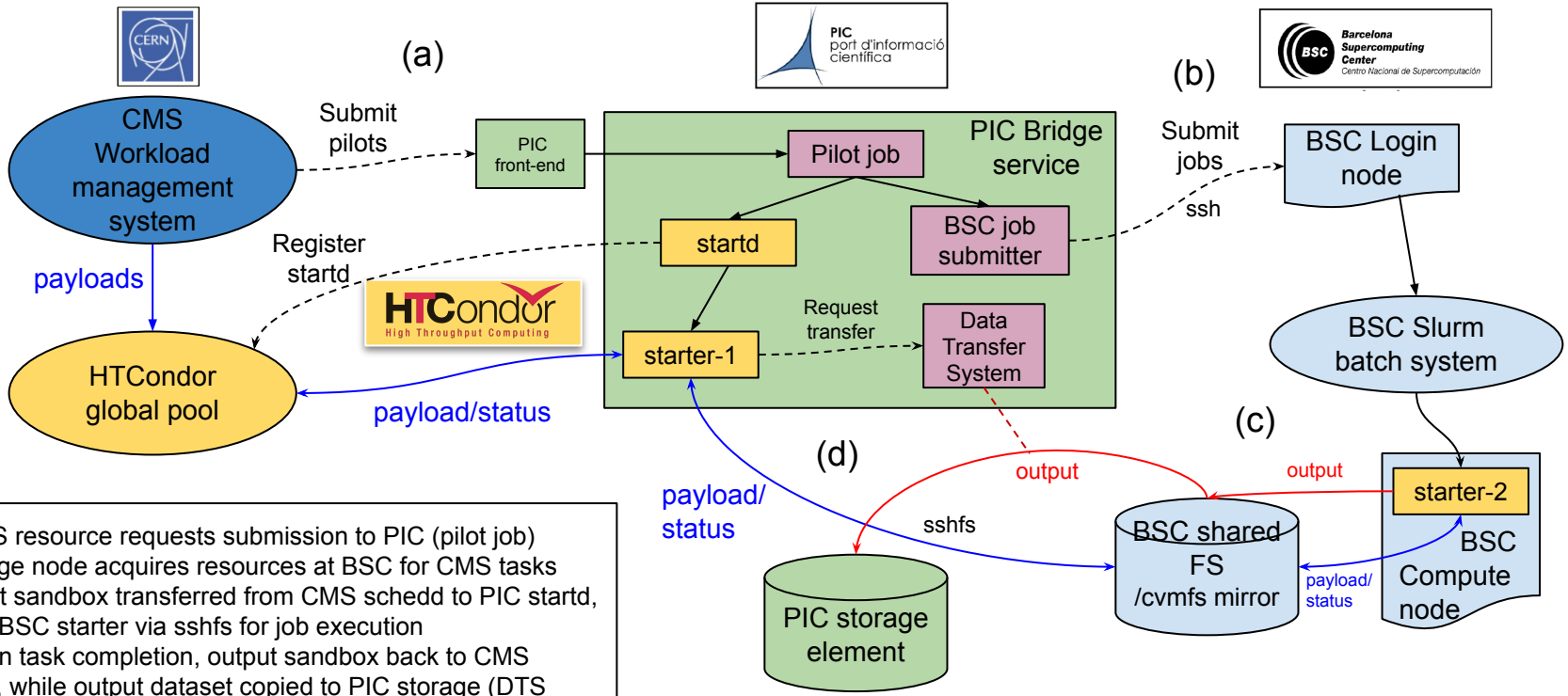
From a functional perspective, **the resources at the BSC node have been registered into the CMS Global Pool**



Other solutions applied

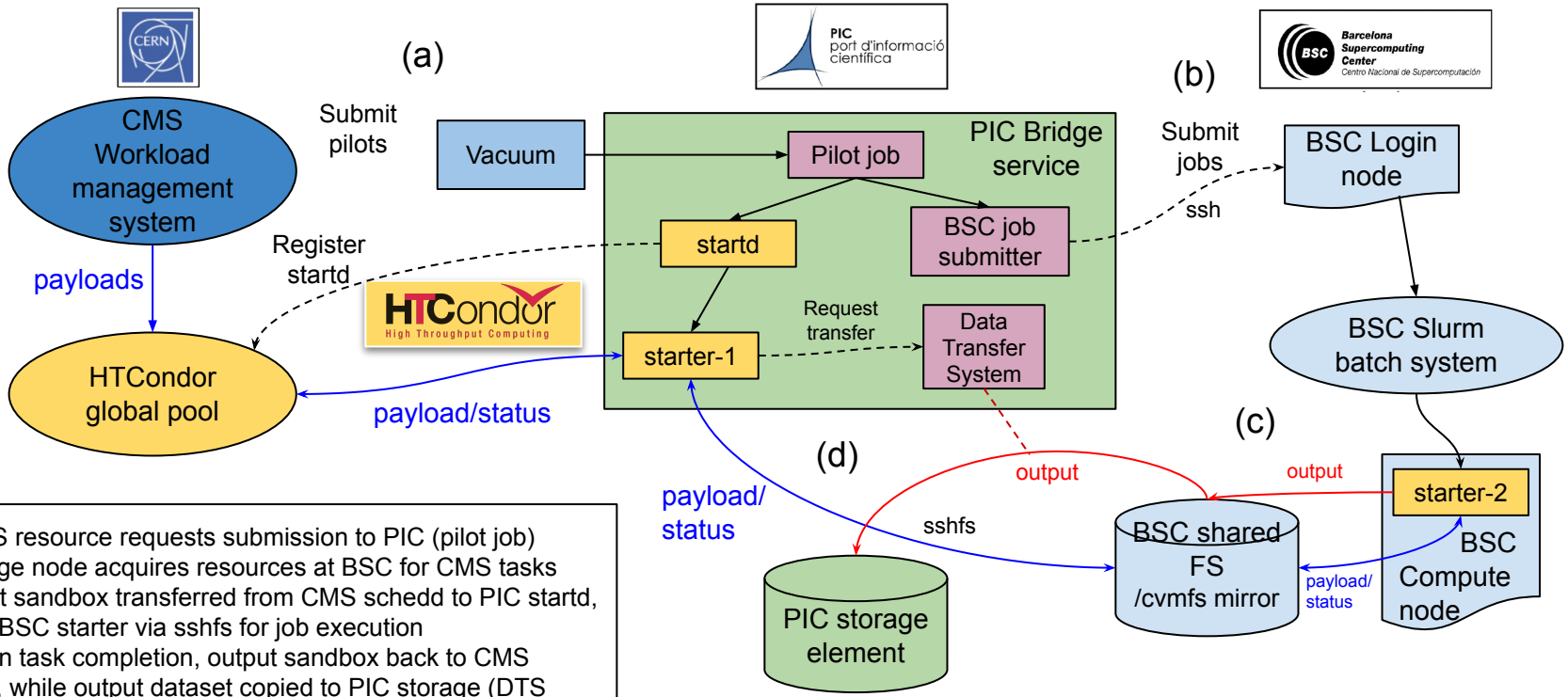
- CMS software (**CMSSW**) deployed to BSC using *cvmfs_preload*
  - Two weeks to copy ~13 TB (37M files) - periodic updates faster (~25 MB/s)
  - Pre-placed some OSG CentOS singularity images (run like Grid jobs)
  - Use cvmfsexec to access copied CVMFS repository for CMSSW
- **Conditions data** read from sqlite files, pre-placed in BSC GPFS
  - Required CMS software modifications - available from CMSSW_11_2
- **Local @BSC environment** configuration for CMS tasks (e.g. where to write output data - aka SITECONF)
- **Develop** and **operate** a **custom data transfer service** (DTS) to transfer output data files from BSC to PIC storage (extending it for stage-in)
- Developing a **new service at PIC** which will start/stop executions if suitable payloads are available at the CMS Global Pool and to control the pilots at runtime to turn the **slurm jobs more efficient**

The current setup



(a) CMS resource requests submission to PIC (pilot job)
(b) Bridge node acquires resources at BSC for CMS tasks
(c) Input sandbox transferred from CMS schedd to PIC startd, then to BSC starter via sshfs for job execution
(d) Upon task completion, output sandbox back to CMS schedd, while output dataset copied to PIC storage (DTS acting as third party copy manager)

The current setup



(a) CMS resource requests submission to PIC (pilot job)
(b) Bridge node acquires resources at BSC for CMS tasks
(c) Input sandbox transferred from CMS schedd to PIC startd, then to BSC starter via sshfs for job execution
(d) Upon task completion, output sandbox back to CMS schedd, while output dataset copied to PIC storage (DTS acting as third party copy manager)

The Data Transfer Service (DTS)

A custom Data Transfers Service (DTS) has been designed and implemented in order to manage **output data transfers** from BSC to PIC storage (“mini-FTS” attached to jobs)

- **Synchronous data transfer:** requests launched by ad-hoc job wrapper
- DTS executes **parallel scp transfers** between BSC and PIC storage
- DTS controls **concurrency** and implements **sanity checks** (size, checksum)
- The transfer is **transparent to CMS WM** services, assume data is already at PIC when job finishes, then proceeds to register the produced dataset to PIC_Disk RSE

Initially, **focus on workflows with no input** (GEN-SIM). The rest of the chain (DIGI-RECO, I/O intensive and needs to read input data files) can then be executed at PIC (or other sites remotely reading those inputs from PIC)

- **Input data** could potentially be served also by a similar DTS, being tested atm

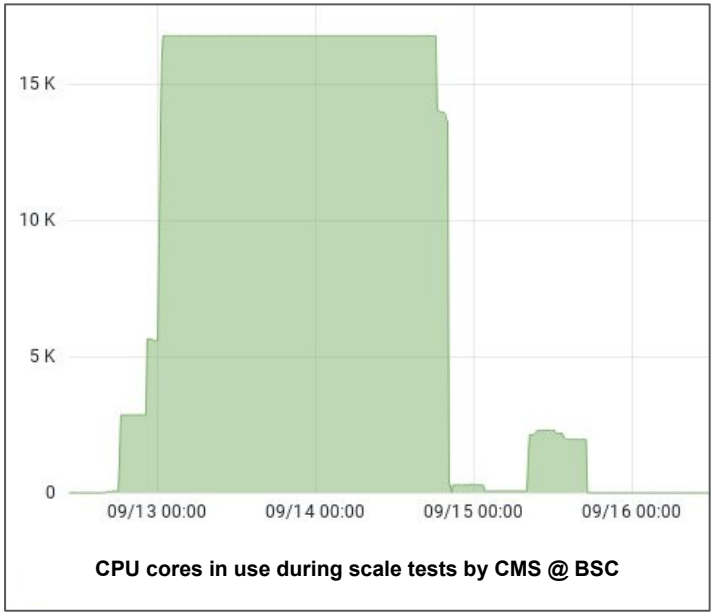
10 Gbps available at BSC. Scale tests showed we can saturate for hours at **800 MB/s**

The Data Transfer Service (DTS)

Monitoring and alarms deployed for the new data service

Scalability tests

A fully working **prototype has been tested at scale** with realistic CMS workloads (GEN-SIM, no input data required) and the infrastructure and services deployed were proven capable to sustain a scale of **~15k CPU cores** in BSC's MN4 (**500 MB/s aggregate output rate**)



CPU cores in use during scale tests by CMS @ BSC

A slurm job takes a whole BSC compute node (48-cores)

Run standard 8-core CMS MC production jobs

Average CPU efficiency for production tasks >90%

A single bridge node at PIC capable to handle this load and the data transfers from BSC to PIC (AMD EPYC 7452. 64 cores (w/HT) - 128 GB RAM - Network: 25 Gbps)

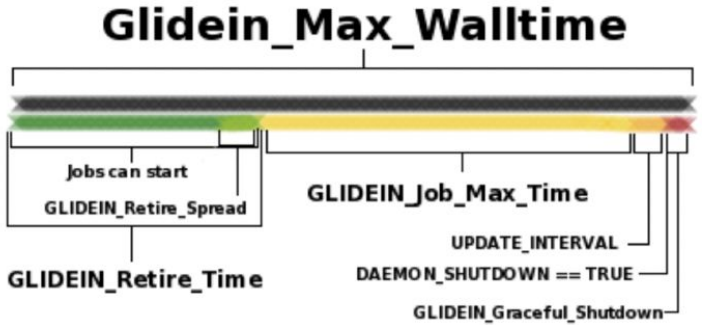
The remaining simulation parts should be executed at PIC (addition of pile-up samples to get the full simulated chain)

Next: supervising jobs at BSC

We need a **'PIC front-end' service** that continuously queries the CMS Global pool to check if there are suitable payload jobs to be executed in BSC, and then estimates the amount of pilot jobs to be sent

This 'PIC front-end' also needs to keep **pressure** at BSC slurm while there are payload jobs to be executed and gracefully **stop sending pilots** when the central tasks are exhausted

Also, slurm jobs take one complete node at the BSC (48-cores partitionable-slot) and we execute 8-core payloads. To improve the slots usage (aka efficiency), we need to **supervise** the slot usages, as a glidein does for CMS

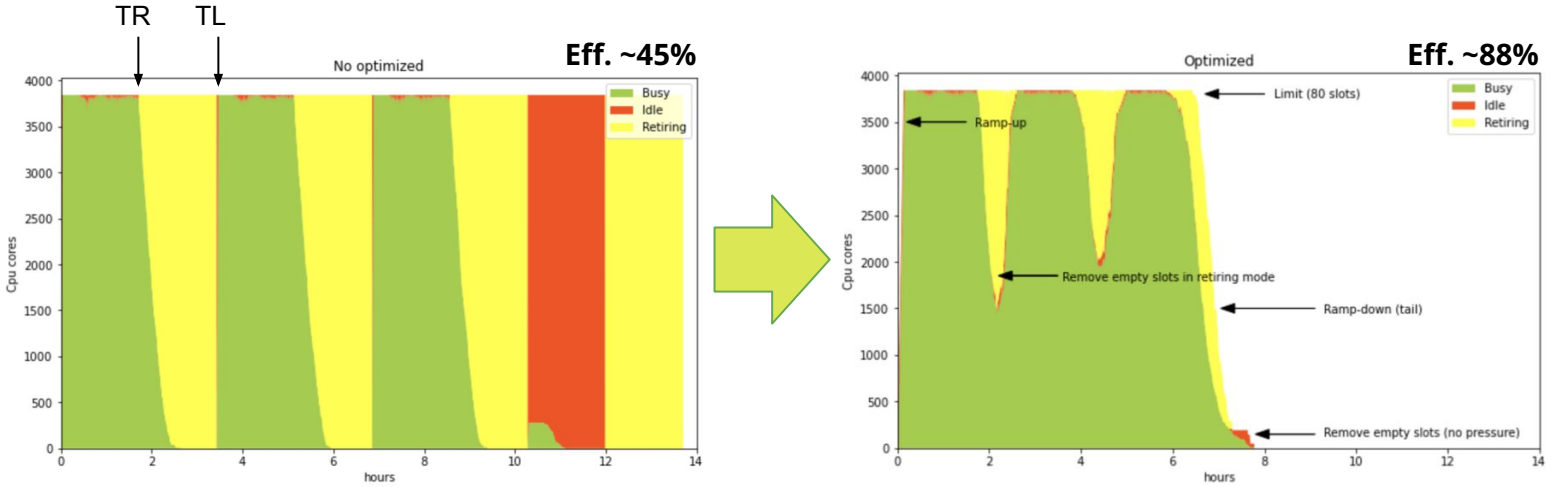


Next: supervising jobs at BSC

CMS pilot slot usage efficiency at Tier-1s is regularly at **~95%**

Developed a 'PIC front-end' service in a PIC testbed which will control the slurm jobs at BSC. We aim at similar efficiency results with the new service, in order not to rely on intensive manual supervision for a reasonable usage efficiency (up to now) - needs to be integrated yet

Features: Remove empty slots in retiring mode / Remove empty slots if there is no payload pressure / Ramp up / Ramp down



Execution of 5k test jobs (payloads) in 80 slots (pilots)

Next: enabling stage-in

Singularity container has an **xrootd-server** which will be used to intercept any of the data reads at runtime

We aim at using this functionality to serve the input data from PIC to BSC, using a already **modified and commissioned version of the DTS that handles input data**

- Either we can inspect a-priori the payloads requirements and send files before-hand, or use the xrootd-server to serve the files at runtime
- Random access to input pile-up samples performed by CMS simulation jobs possibly not covered by this service

E.g. enabling **RAW data reprocessing** at MN4 would be possible (testing it atm)

Next: other ‘nexts’

Expected enhanced **network** connectivity between BSC-PIC for upcoming MN5

- PIC already at 200 Gbps. BSC expects 100 Gbps along 2023 (current 2x10 Gbps)

Improving **monitoring**, **alarms** and implementing **automated recovery** for the services deployed

Optimize resource **usage efficiency** with the new **‘PIC front-end’ service**

Automated **accounting** algorithms to properly report BSC usage by CMS

Address **(minor) communication** with central services (monitoring, log collect,...)

Summary and Conclusions

The **HTCondor via shared filesystem** idea has been developed by the condor team and implemented at PIC

Functionality and **scale tests** have been run, linking the BSC resources with the CMS (ITB) Global Pool

- SIM workflow executed manually from the CMS ITB schedds and the outputs transferred into PIC storage using the DTS

The infrastructure and services deployed were proven capable to sustain a scale of **~15k CPU cores** in BSC's MN4 (~10% of BSC!) and **500 MB/s aggregate output rate**

We need to **consolidate the model** and further integrate it into CMS WM and operations. More workflows would be suitable when the **stage-in** feature will be commissioned

Big effort so far!

Thanks!

Questions?

The HTCondor split-starter (details)

In order to make use of the core component of HTCondor communication via FS, a number of steps must be performed along with each slurm job submission

Enable a set of algorithms to perform these actions, creating a “mini glidein”, which can be later on implemented with proper GlideinWMS glideins

Actions:

- Pick unique rendezvous directory in BSC GPFS per job
- Setup sshfs mount at PIC bridge
- Submit BSC slurm job
- Configure and launch startds at PIC when slurm job starts
- Wait for BSC job to exit, then
  - shutdown startds
  - remove sshfs mount (clean up remote directory first)
  - clean up functional dirs at PIC

The Data Transfer Service (DTS)

- Tested **single stream multi-file-copies** and **multithreaded** methods (scp's, cp's, globus-url-copy).
- Transfers from BSC **saturating** at ~800 MB/s
  - On a **10 Gbps network link**, so about ⅔ of the max theoretical capacity in use
  - **Similar saturation** regardless of the copy mechanism
- For simplicity (e.g. no extra authentication step), **scp** selected as the DTS method to copy files from BSC GPFS to PIC dCache.