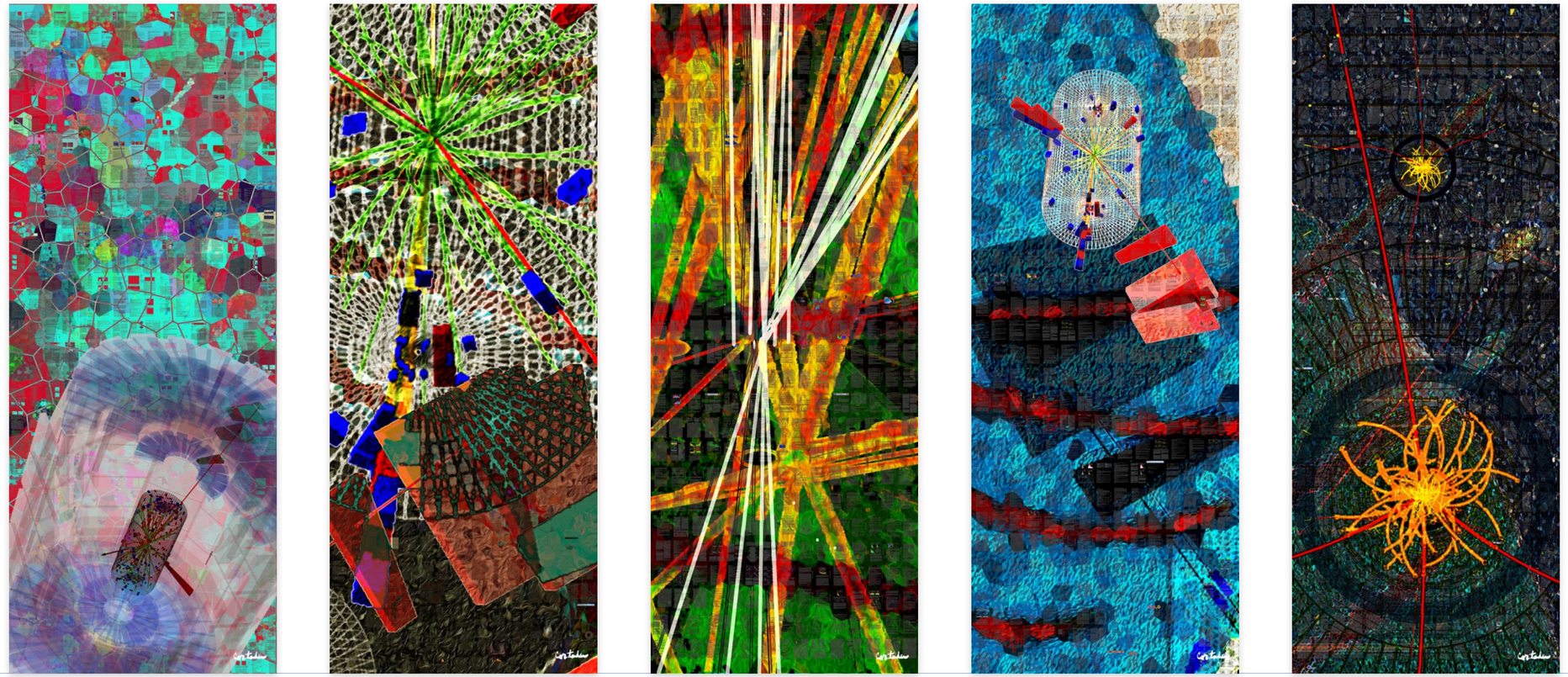
CMS Perspective on WM

Eric Vaandering

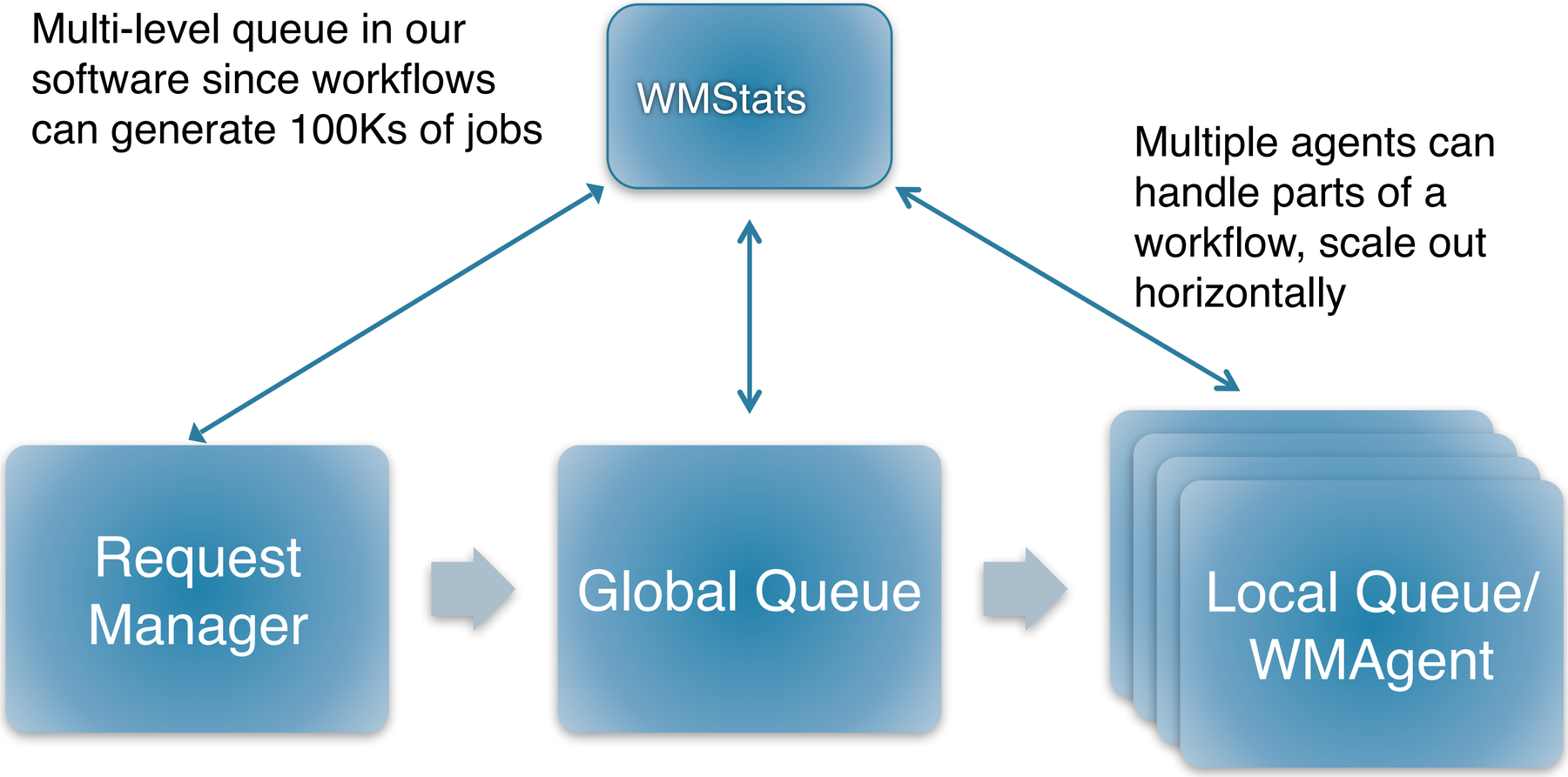


Intersection with Data Management

- It's difficult to discuss workflow management without saying at least something about data management
- The CMS model is to send jobs to data (most of the time).
  - We have way too much data/too many sites (~60) to ignore this
  - Very few replicas of most data
- Primary data management consists of
  - DBS (dataset/file catalog, runs, sub-runs, other meta data)
  - PhEDEx (location metadata and movement system, subscriptions)
    - ★ Uses FTS underneath
    - ★ Replacing with Rucio in 2019-20
- Secondary data management:
  - AAA/xrootd: remotely readable global namespace
  - Dynamic DM: issues PhEDEx commands to replicate(delete) (un)popular data
    - ★ Also to be replaced with Rucio
  - Exploring regional caching as well (based on xrootd)

WMAgent diagram

Multi-level queue in our software since workflows can generate 100Ks of jobs

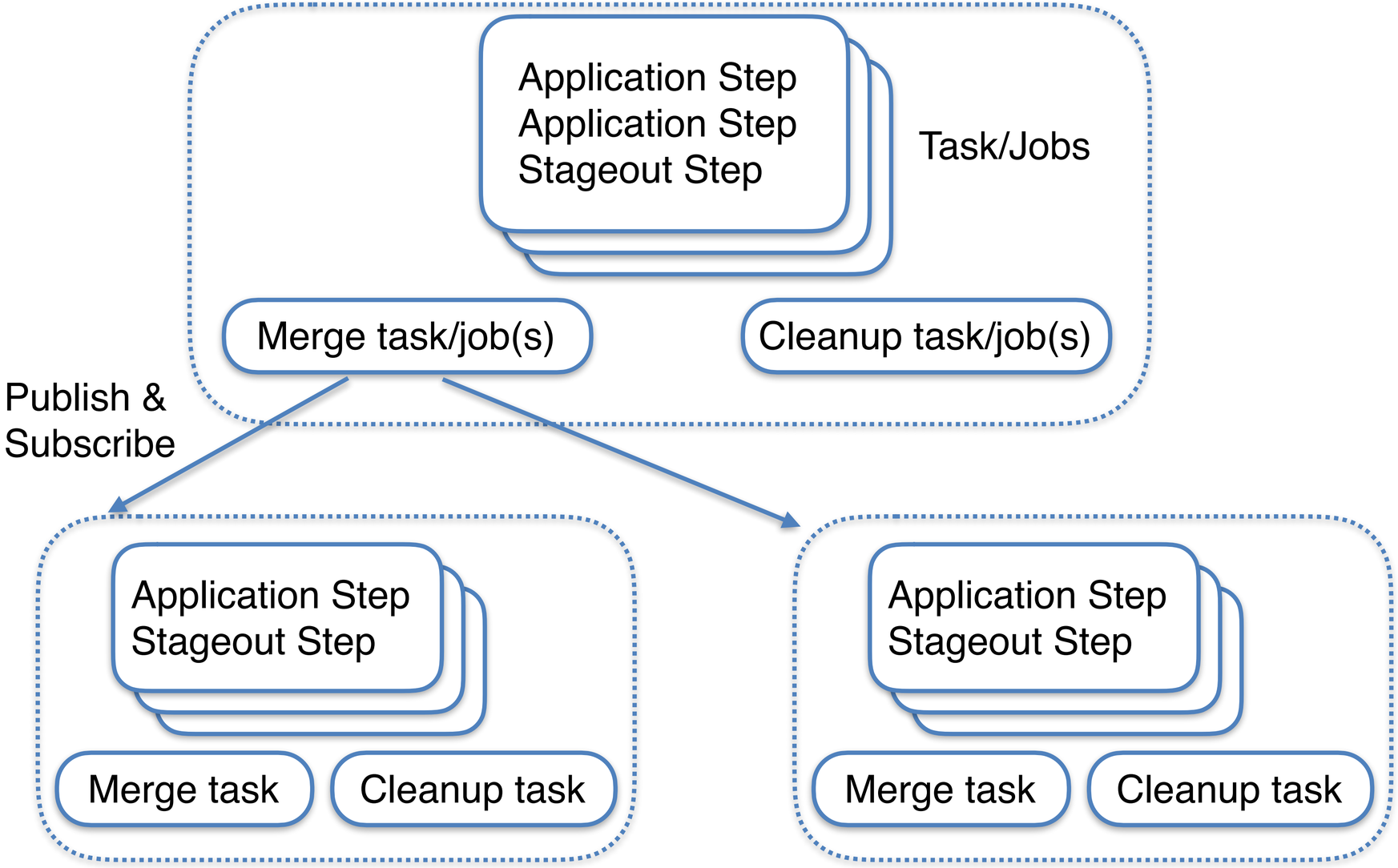


Multiple agents can handle parts of a workflow, scale out horizontally

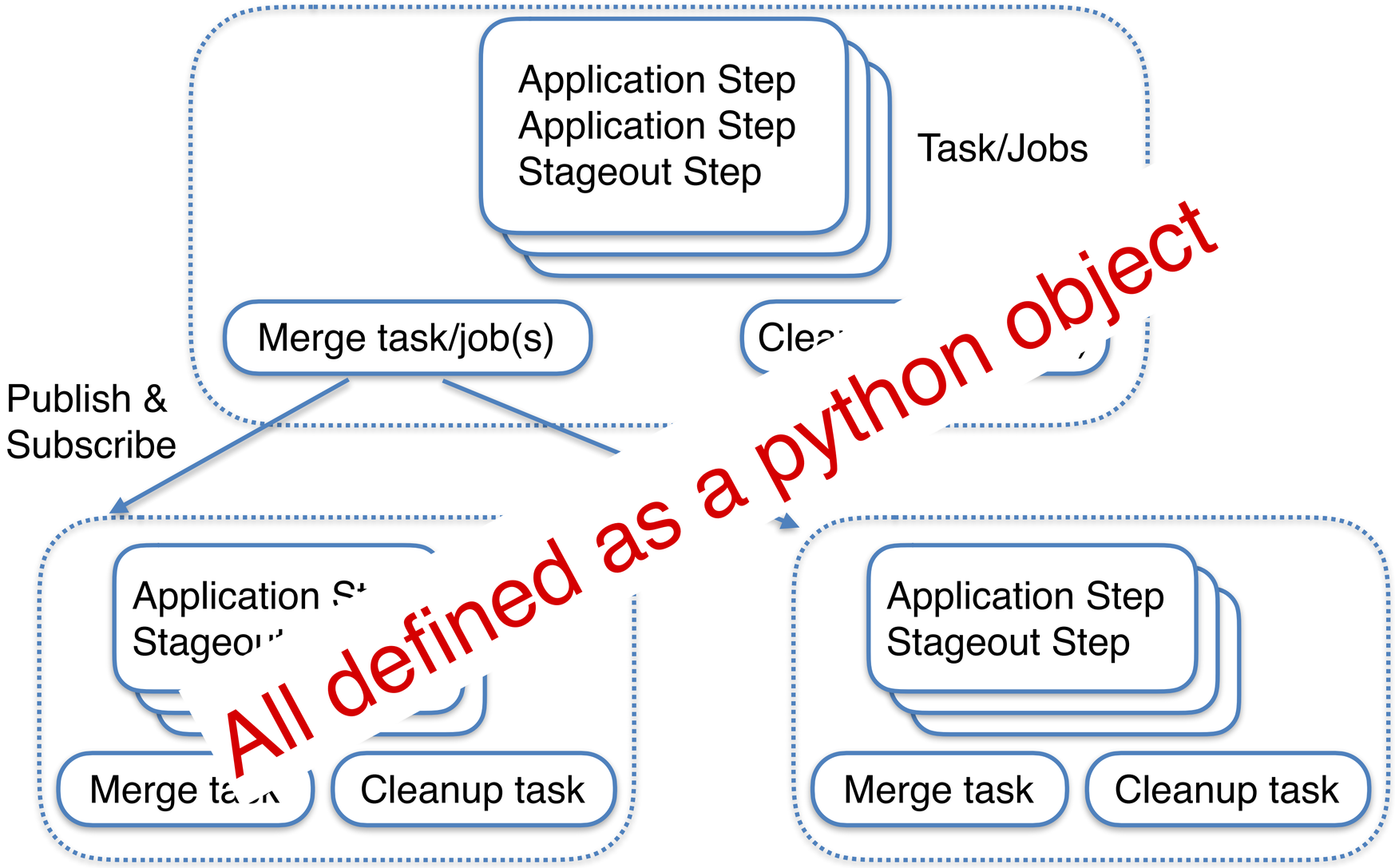
Workflow Flexibility

- WMAgent has a lot of different ways to complete complex workflows
  - CMS processing is typically multistep in a single workflow. Example:
    - ★ Step 1: produce generated (particle level) MC and simulated (hit level) MC
    - ★ Step 2: produce digitization simulation
    - ★ Step 3: produce reconstructed data
    - ★ Step 4: produce analysis formats (miniAOD, nanoAOD)
    - ★ May or may not want to keep outputs of each of these stages
    - ★ Output from each step is merged to O(GB) files easily stored on tape
  - Different ways to accomplish this
    - ★ Can run all steps of the process in a single job
      - Advantages: easy book keeping for resubmission, reduced # of jobs, reduced merges, can be better matched to HPC/Cloud
      - Disadvantages: all code has to be ready at the same time, compatible requirements for each step, parentage after merges
    - ★ Can run each step in a separate set of jobs
      - Advantages: see disadvantages above, possible to run steps on different hardware resources (typically not), still asynchronous
      - Disadvantages: see above, may merge and store data not ultimately kept, illusory flexibility in job locations
  - Designed to have processing subscriptions on growing datasets
    - ★ Not exercised, CMS did not need this

Sample Workload



Sample Workload



Aside on Pegasus

- Naively you'd expect that the description of multi-step workflows would map well to Pegasus (based on Condor DAGMAN)
- We evaluated it and it's not well matched
  - Our merges are based on target file size — no idea how many step 1 jobs feed output to a merge job
  - We don't know where are jobs will run until *after* they start
    - ★ Job 1 and Job 2 of Step 1 may write data at different sites → inputs to different merge jobs
  - No real need for headline Pegasus feature: automated data movement

Dependent on CMS services

- Components of WMAgent communicate with a number of CMS services (all REST based)
  - SiteDB/Dashboard for understanding grid configuration/site status
    - ★ Migrating to CRIC (nee AGIS), a common WLCG project
  - DBS/PhEDEx for data discovery (what data is in a dataset, where is it?)
  - Components that publish data into DBS and subscribe data to their final destination(s)
  - Have or planning to change out or upgrade all these layers with minimal disruption

WMAgent interactions

- Tier0 (near real time processing) is a specialized flavor of WMAgent

- Resource provisioning and job execution delegated to HTCondor/GlideinWMS
  - DAGs are interesting, but not really flexible enough for our workflows
  - Tell GlideinWMS all the places a job can run, resources needed, it takes care of the rest
  - Part of “rebrokering” is handled by GlideinWMS: jobs waiting can be overflowed to other sites well connected (xrootd) to the data
    - ★ Other way is that new locations from DDM can be included before jobs are submitted to GlideinWMS
    - ★ Interest in directly declaring data needed as a job requirement
  - Plans to enforce overall job limits within GlideinWMS
    - ★ e.g. merge jobs are hard on sites, need to limit the overall # running per site
    - ★ currently managed by restricting number submitted *per agent*
  - Resubmissions handled by agent based on return codes (some retried, some not)

Client services

- Workflow planning and checking was major operator overhead (1000s of simultaneous workflows)
- External services and scripts feed work into Request Manager via REST interfaces
  - McM and Unified used to construct workflows and pre-stage data
  - Back end checks prior to announcing data is ready, preparing recovery workflows
    - ★ aim to vastly reduce the recovery workflows in next couple of years by incorporating into WMAgent
- Request Manager permanently holds request information which can be aggregated with dataset metadata

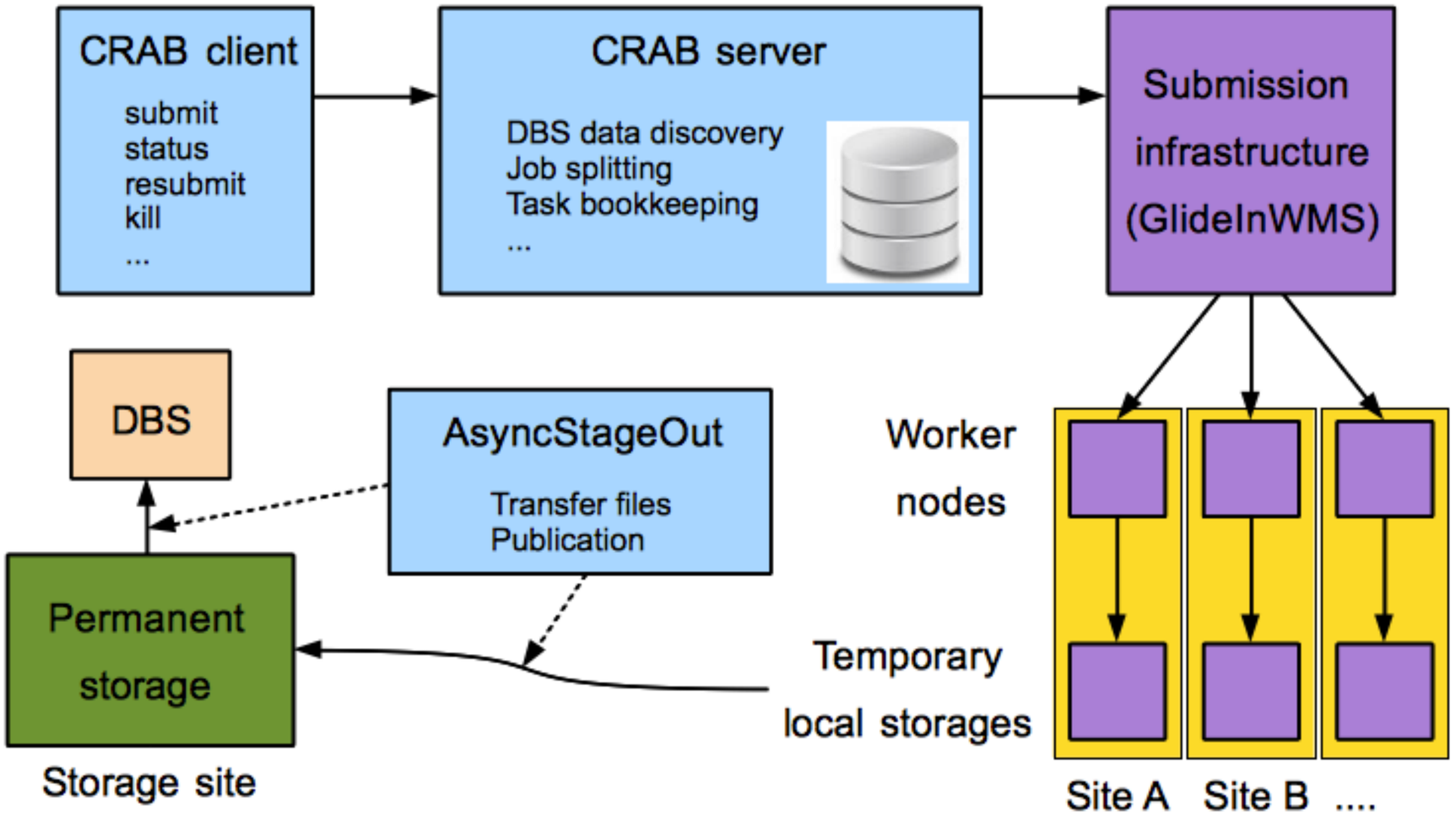
Recovery

- Recovery workflows are generated, parameters of the workflow can be changed, and workflows resubmitted to catch up missed work
- The need for these workflows and their frequency is a major pain point for us
  - Can only be initiated after main workflow is closed out
  - Can be complicated in the different steps in different jobs case
    - ★ Do you try to re-read needed data or regenerate it? Does it still exist?
    - ★ Parts of the output from several input jobs may be needed as input for one subsequent job

Analysis system

- Second system, similar in design to JobSub, for user analysis
  - Some underlying code shared with WMAgent
  - Also reliant on GlideinWMS plus some use of DAGs
    - ★ Jobs go to same global GlideinWMS pool for prioritization
- Differences with production system
  - Package and ship user code to worker nodes
  - Simpler workflows, better status tracking
  - User client driven — more interactive
  - No merging (yet) of outputs
  - Uses a different data movement system (also based on FTS)
    - ★ Will also be moved to Rucio

CRAB Architecture



CRAB tasks

- User-friendly way to accomplish all needed steps of an analysis
  - Data discovery (what's in my data and where is it)
  - Job splitting (each job runs on a reasonable portion of the data)
    - ★ Atomic unit in CMS is a luminosity section, 23s of data
  - Configure and run CMSSW (cmsRun) to run on correct files, lumis
  - Submit jobs
  - Publish resulting data in data catalog (DBS)
  - Move data to users' "local" institution (ASO and FTS)

Scale

- Currently keeping around 200k cores busy
  - Most processing done with multiple core jobs (up to 8)
  - Most jobs still single core
- About 2M jobs/day
  - More or less 1/2 production, 1/2 analysis
- Scale out by horizontal scaling HTCondor Schedd
  - CRAB — Multiple schedds, one task worker
  - WMAgent — One agent per schedd
    - ★ Limit from local MySQL database is similar (supports MySQL and Oracle)
- Also scaling limits in GlideinWMS Global Pool — beyond what we need now, but being addressed for the future

CMS Workflow Management Review

- A review to determine if CMS is on the right path. Three questions (and likely answers):
  - Is CMS WM ready for HL-LHC?
    - ★ Most components scale OK (horizontal scale-out or not limited)
    - ★ Heterogeneity of resources is a problem, especially related to target job lengths
      - HPC/Opportunistic resources can be best used on “few” hour scale vs. days for grids
    - ★ Not flexible enough now for the constellation of HPC resources (especially connectivity)
    - ★ Evolution of existing tools will be sufficient
  - Is the split between production and analysis systems justified?
    - ★ Clearly many of the concepts are the same
    - ★ Should explore the same backend, but different frontends (again)
      - Users have different expectations in “interactivity”, user friendliness, quick WF execution vs. sheer amount of work
  - Are there developments outside of CMS that should be considered as a base for workflow management?
    - ★ Will be no recommendation to throw out what we have now
    - ★ Acknowledges that WMAgent arose to solve CMS domain problems. Panda, DIRAC, Pegasus have different emphases
- Review will conclude this year (next session later this month)
- My take: Still interested in common projects to abstract WF dependencies

Backup

- Backup slides

Rough estimate of development effort

- WMAgent initial development 2009-2011:
  - Includes Tier0, which is WMAgent based
  - 6.5 FTE/yr, 20 FTE-yr total
- WMAgent maintenance 2012-2016
  - 3 FTE/yr, 15 FTE-yr total
- GlideinWMS development
  - 2006-2009: 1.5 FTE/yr
  - 2010-2016: 2.5 FTE/yr
  - 20-25 FTE-yr total for all stakeholders, not just CMS
- CRAB development effort 2013-2016
  - Another ~15 FTE, now ~2 FTE/yr maintenance
- Operational effort for CMS
  - ~3 FTE for WMAgent
  - ~3 FTE dedicated to GlideinWMS

CRAB3 Condor/Glidein Interface

- We make light use of DAGMan and heavy use of Glideins
- DAGMan is used to separate tasks into job running and monitoring of data transfer, publication
- Glideins to limit execution sites, resources, etc.
  - US operates in failover mode — jobs waiting for some time redirected to other US sites, data streamed over xrootd

WMCore package

- Request Manager
- WorkQueue
- WMAgent
- WMStats (monitor)
- ACDC Server
- (T0- build on top of WMAgent, T0_WMStats)
- (DBS, CrabServer, DAS, SiteDB) – uses some WMCore library

What it does

- Help operation (monitor progress, trouble shooting, etc)
- Take request (workflow specification)
- Create jobs
- Submit jobs (to batch system, GlideIn/Condor)
- Track jobs, Retry jobs (job level, workflow level)
- Monitor jobs (by workflow)
- Archive workflow summary
- Archive data/statistics (outside the system – DBS, PhEDEx, Dashboard)