Large scale job management and experience in recent data challenges with in the LHC CMS experiment.

Stuart Wakefield, Imperial College London.

On behalf of the CMS production developers

Outline

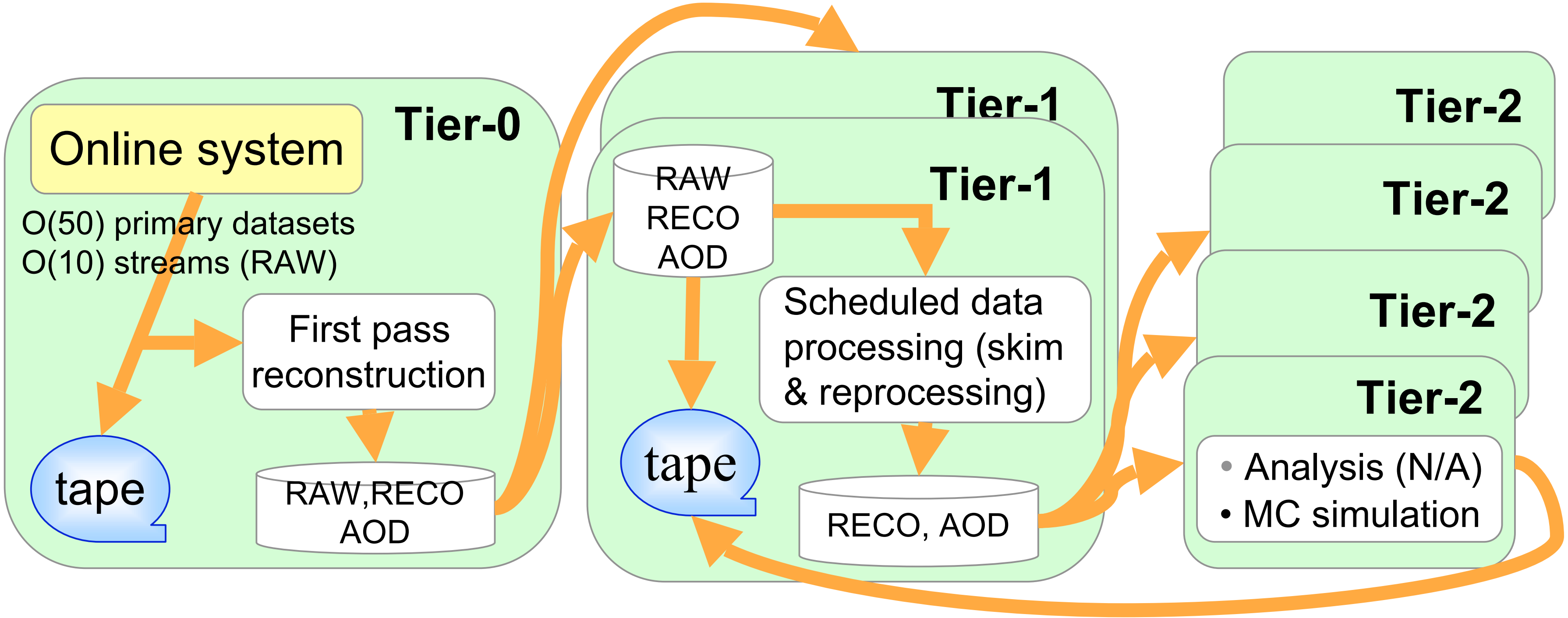
- Introduction.
- CMS production and processing.
- Current architecture.
- Data challenge and real data experiences.
- Future Plans.

Introduction

- CMS data production/processing requirements incl.
  - 2008: 1000M+ events so far (incl. FastSim + challenge data)
  - 2009: 1280M RAW and 1280M MC.
- Distributed resources (1T0, 7T1 and 30+ Tier2)
  - Submission via grid resources (LCG, OSG, ARC) or local batch system
  - Varied storage technologies
  - Sites with the same technologies still have significant differences

Data activities

- Low latency critical processing (See Tier0 talk tomorrow):
  - Prompt Reconstruction
  - Express Stream
  - Alignment and calibration (AlCa)
  - Data Quality Monitoring (DQM)
- Offline processing:
  - Reconstruct * 3 (2009)
  - Dataset skims of events with set criteria (after each Reco step)

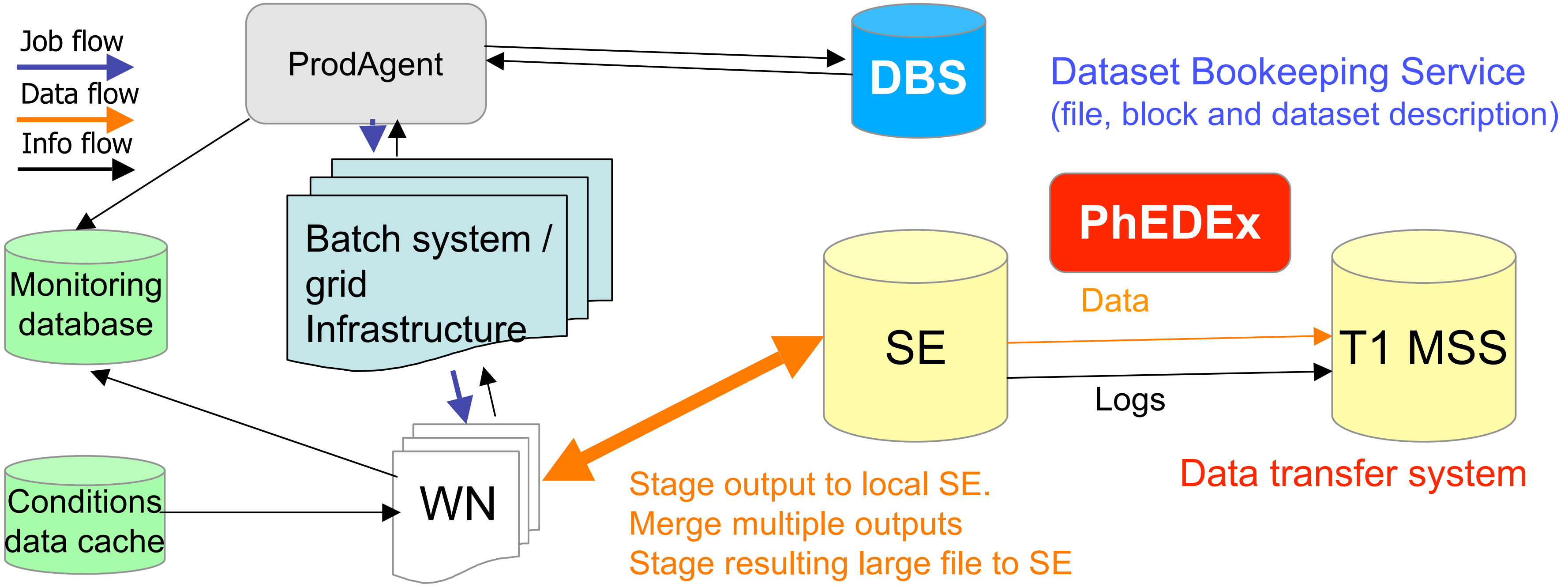


ProdAgent

- Workflow management used by all organised cms processing.
- In use for 2+ years
  - more recently adopted as base for Tier0 quasi-real time data processing.
- Automation
- Scalability
- Highly configurable/extensible:
  - Production and Processing, Real and MC, Online and Offline.
  - Grid and Non grid
  - Work with different site setups (storage, batch system)

General workflow

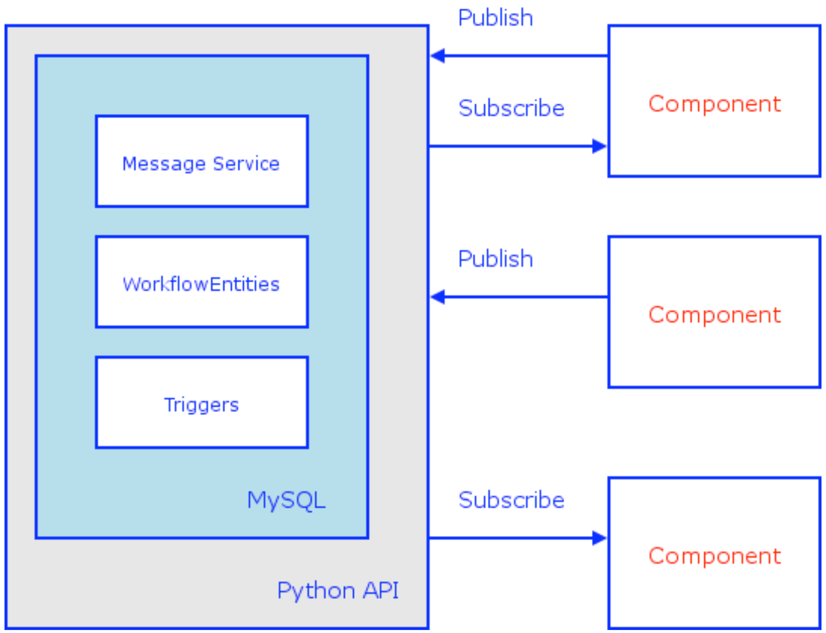
- Jobs limited to contacting local services:
  - Site SE
  - Site conditions database cache
- Small products merged at site
  - Intermediates deleted asynchronously after merging
- Update DBS/PhEDEx asynchronously



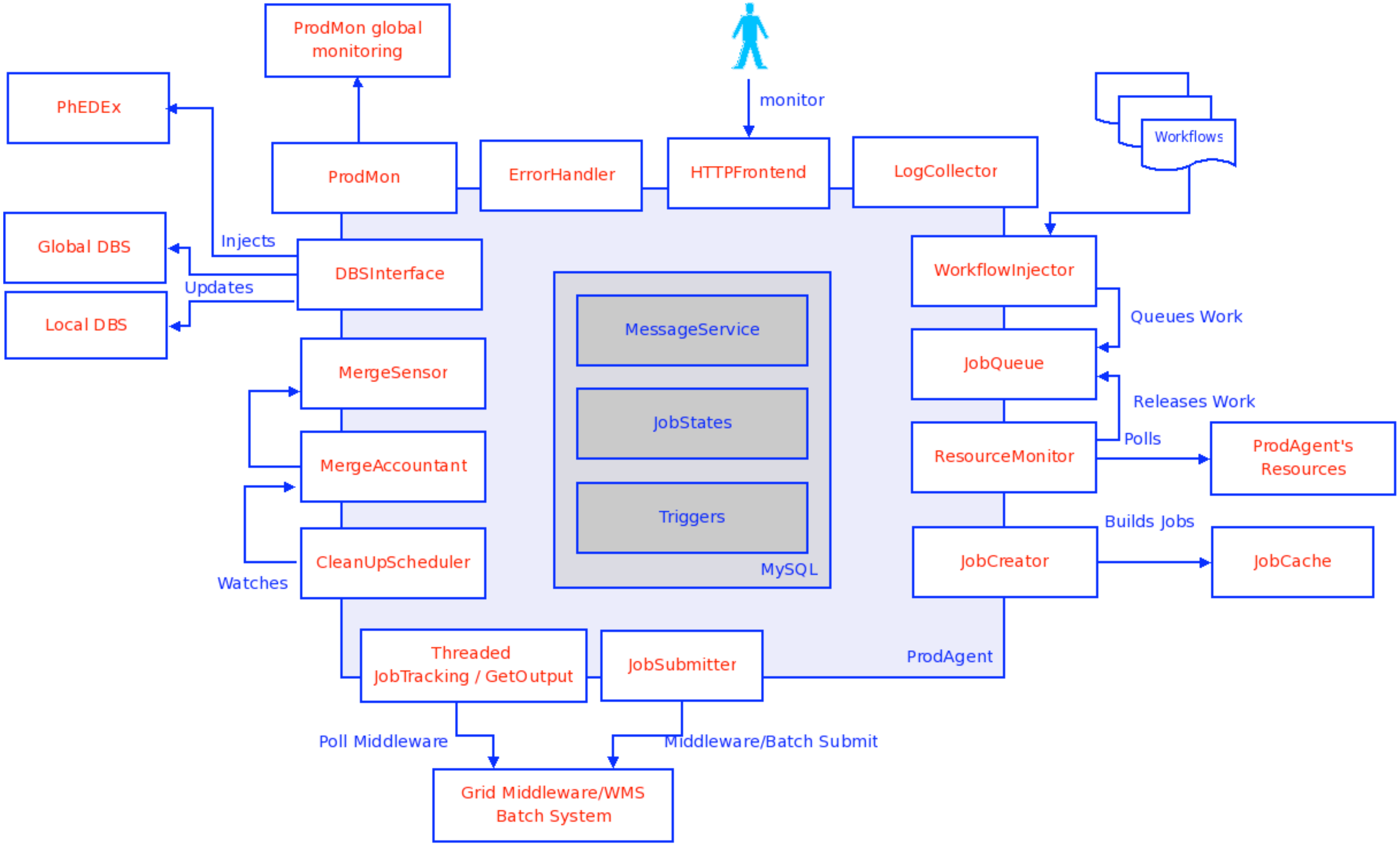
Stage output to local SE.
Merge multiple outputs
Stage resulting large file to SE

Architecture I

- Independent components working asynchronously
- Written in Python - low entry barrier for developers
- Local MySQL database.
  - Persistency
  - Communication between components
- Components call plugins when specialised is behavior required

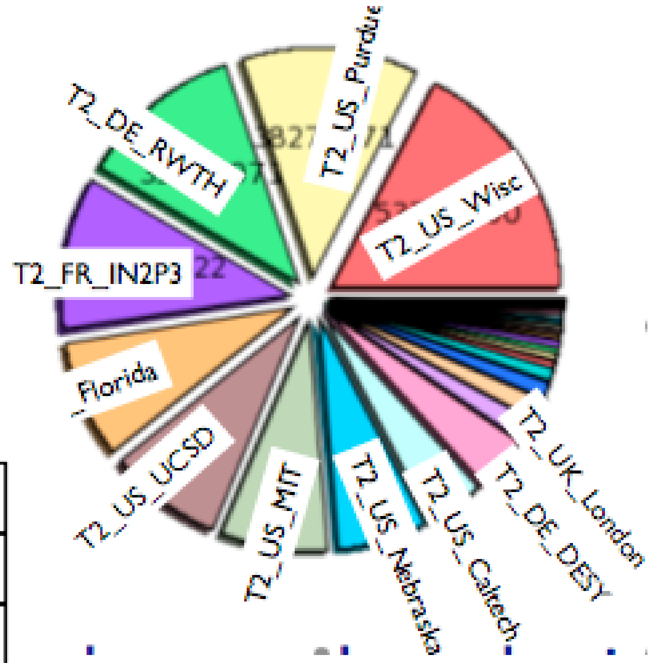


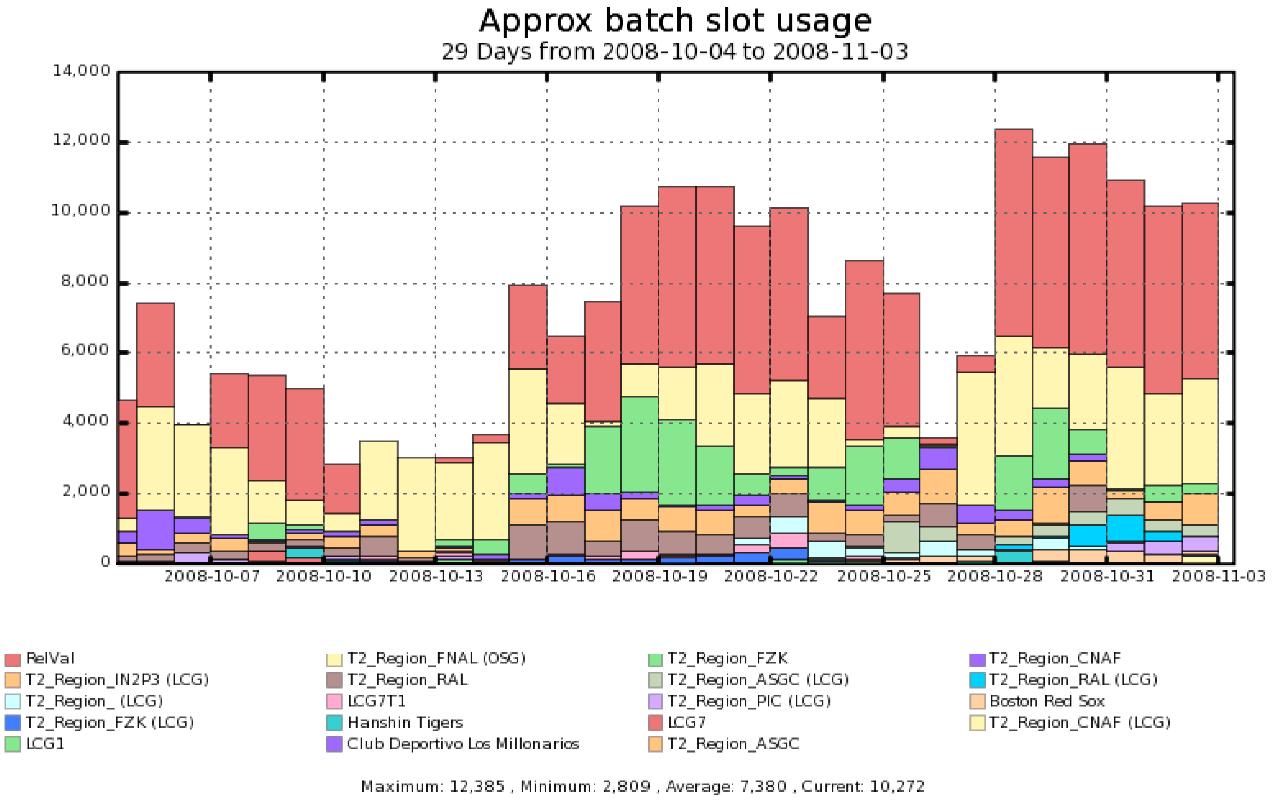
Architecture II



Recent experiences I

- Recently reached nominal startup goals
  - 100M events a month
- Utilised multiple prodAgents to reach goal
  - Only 1 submission technology per prodAgent.
    - Using (parts of) grids
    - Individual site Batch systems



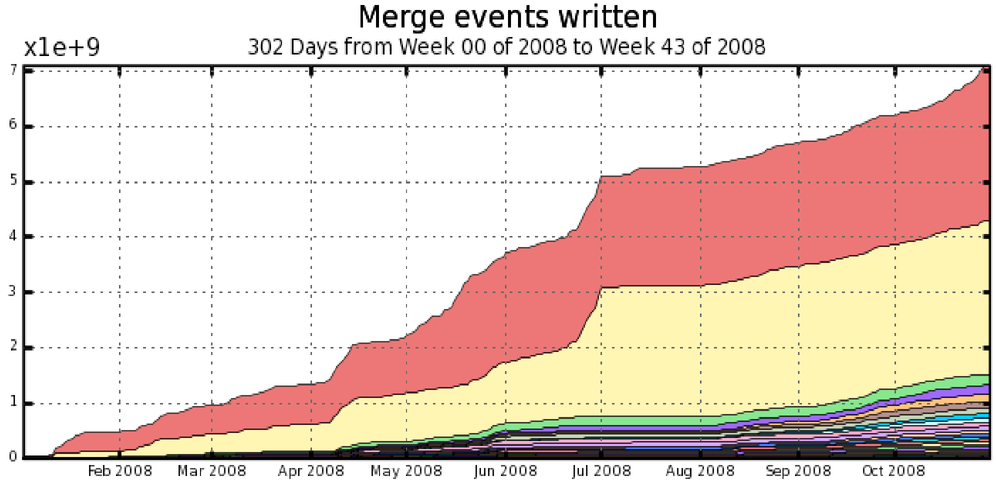

Approx batch slot usage

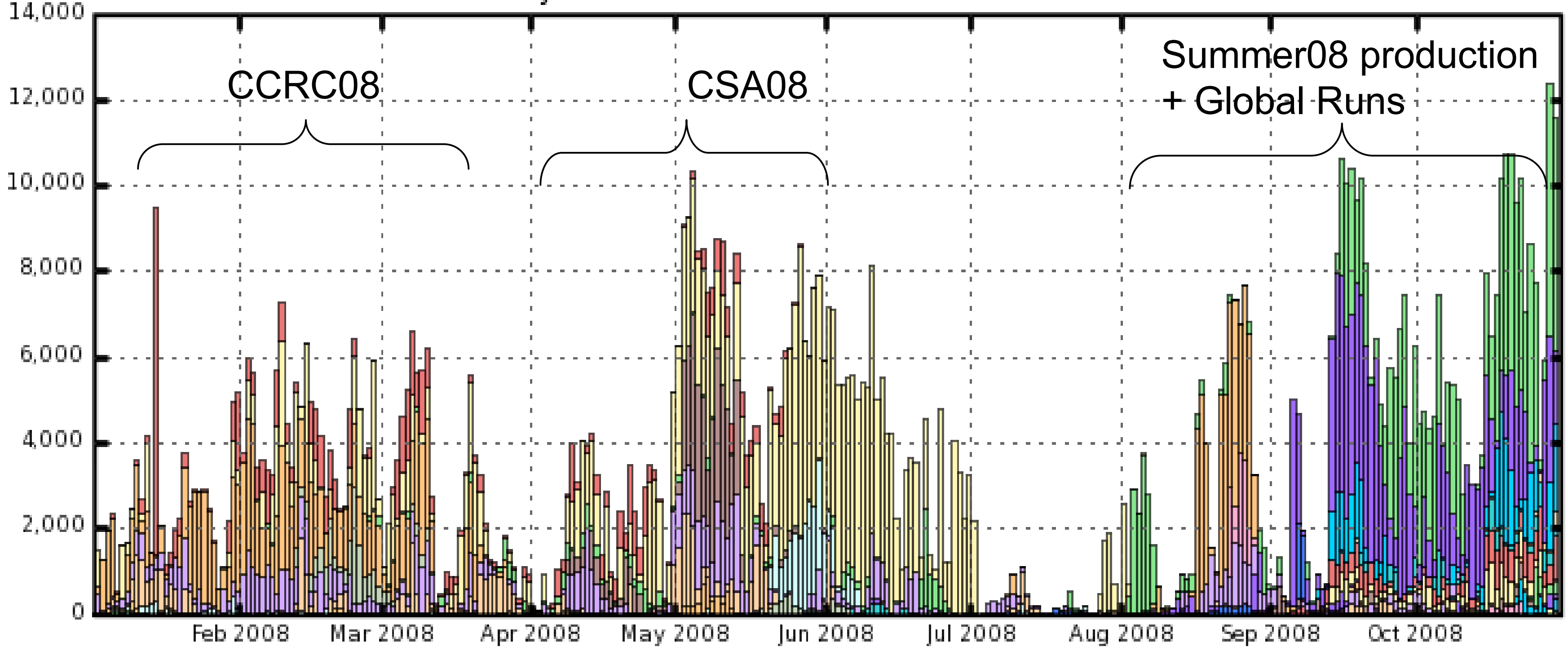
29 Days from 2008-10-04 to 2008-11-03

Maximum: 12,385 , Minimum: 2,809 , Average: 7,380 , Current: 10,272

Recent experiences II

- Recent use includes:
  - Computing challenges
  - Cosmic runs
  - Beam Runs
  - MC activities





CCRC08

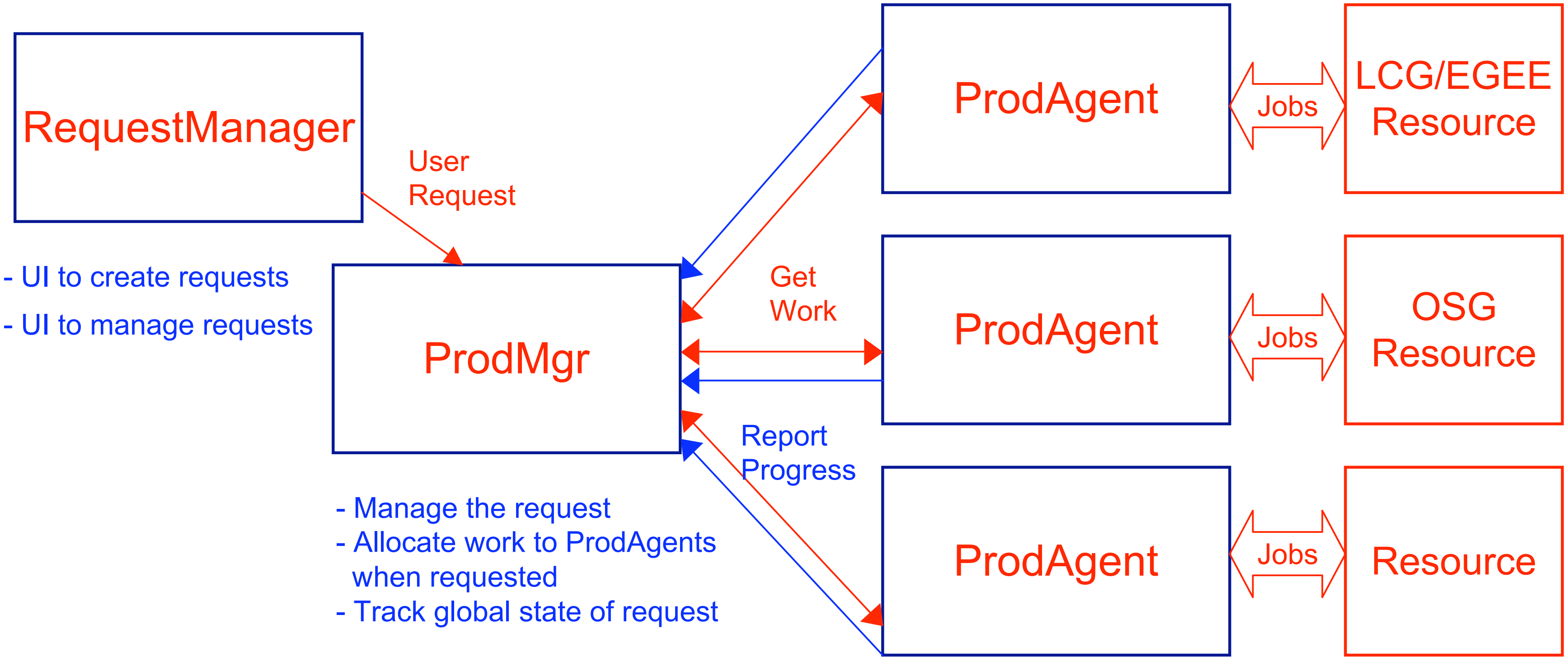
CSA08

Summer08 production + Global Runs

Development plans I

- Increase resource usage
  - Enforce testing of workflows before running
  - Automatic distribution of work to ProdAgents
- Improved operator / physicist feedback
  - Manage workflows (approve, priority etc.)
  - Improve operator monitoring
- Improve scalability
  - 6500 batch slots under one prodAgent (local batch submission)
  - ProdAgent spread over multiple nodes
  - Increase components throughput
- Speed development

Development plans II



- UI to create requests
- UI to manage requests

- Manage the request
- Allocate work to ProdAgents when requested
- Track global state of request

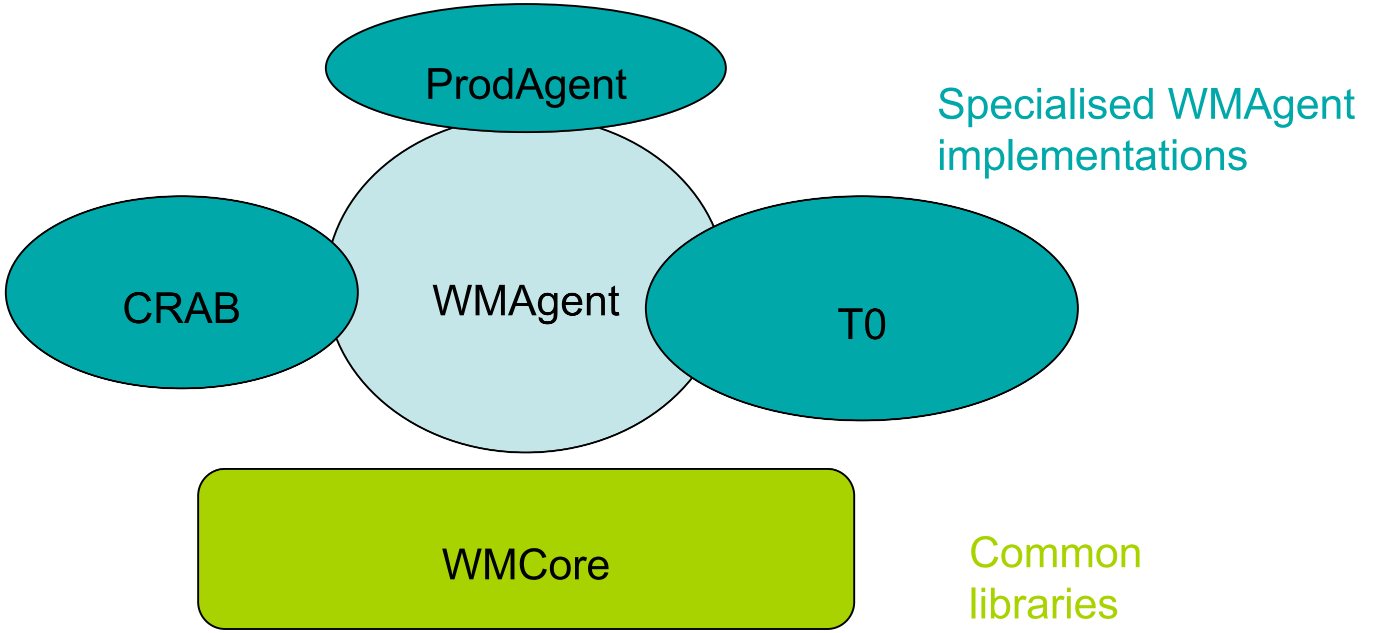
- Convert work into processing jobs
- Create, submit & track jobs
- Manage merges, failures, bookkeeping

CPU + Storage

Possibility for small scale “test” user facing request service for small requests.

Development plans III

- ProdAgent now used in MC production, processing Tier0.
- Large amount of duplicate functionality with CRAB (analysis)
  - Crab server follows Component model
- Move common library functionality to common area (WMCore):
  - Code review + testing.
  - Remove duplicate code.
  - Provide: Job/workflow definitions, job submission, dbs etc.)
- Move common Agent functionality to common layer (WMAgent)



Development plans IV

- Scaling issues for some components:
  - Threading (JobTracking, GetOutput) (See Crab talk)
  - Buffering into Bulk operations (DBS)
  - Remote messaging service
    - Allow distributed components
      - Improve scalability
      - ReqMgr/ProdMgr communication
- Task Queue
  - Pull in work appropriate for the job
    - Available data (site and local disk)
    - Worker node/queue attributes
  - Needed by T0, ProdAgent to take advantage
  - Could combine with pilot jobs
- Monitoring
  - Current ProdAgent monitoring labour intensive
  - Web frontend exists but needs to be extended
  - Alerts for error conditions

Conclusion

- CMS is making use of a performant reliable production and processing system.
- This system has already been shown to scale to startup needs.
- Further work is needed to increase its scalability for increasing data volumes and to reduce the operational load.
- https://twiki.cern.ch/twiki/bin/view/CMS/ProdAgent