Harvester and Yoda.
Next generation of PanDA and ATLAS software products

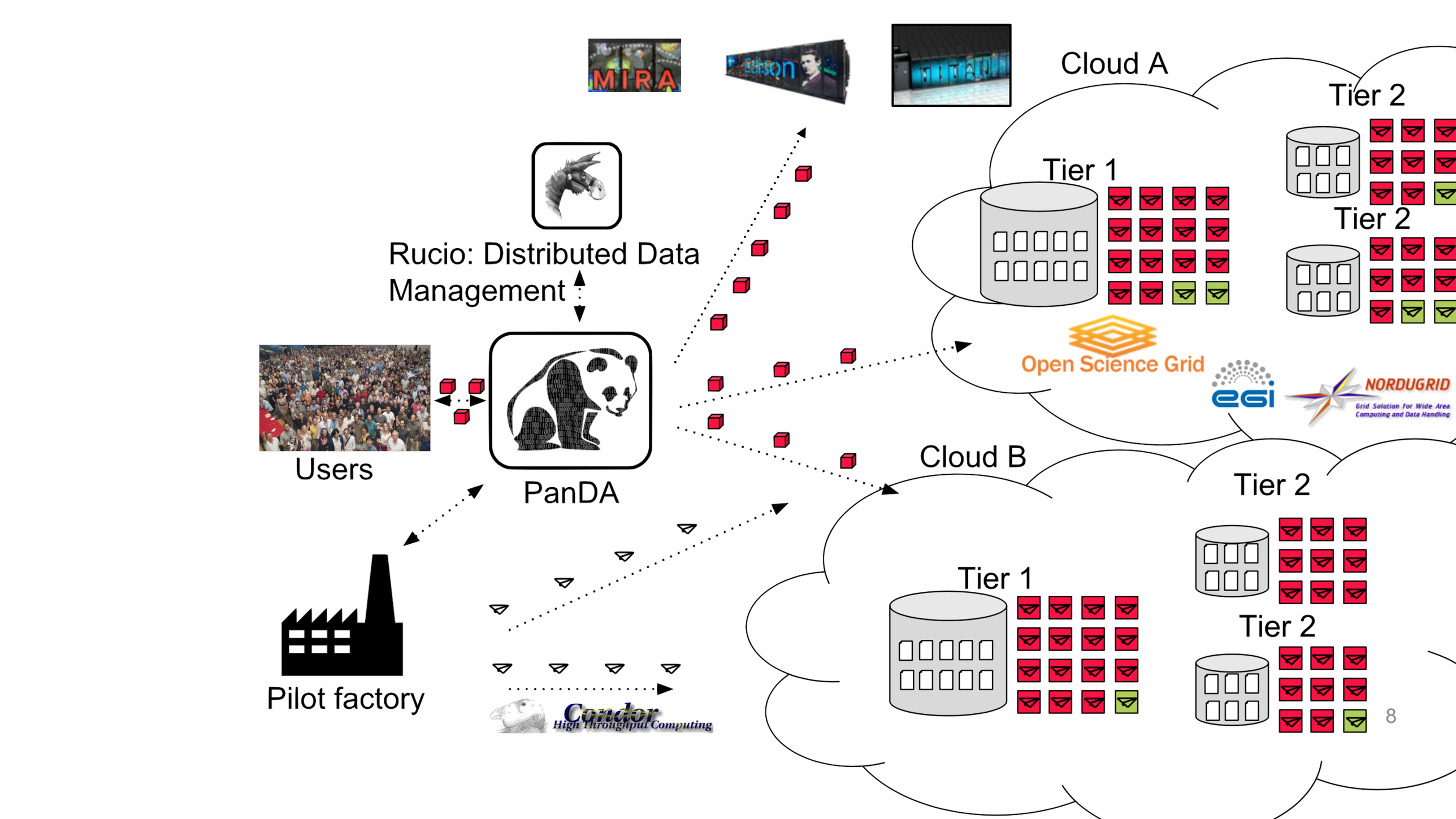
Danila Oleynik
UTA

Outline

- Introduction
- Harvester: resource oriented application in PanDA
  - Harvester design
  - Current status and nearest plan
- Yoda: fine grained processing of ATLAS data on HPC
  - Development stages
  - Yoda design
  - Current status and nearest plan

Introduction

Last few years PanDA ecosystem starts growing and about one year ago was decided to reimplement PanDA Pilot from scratch. It was clear that PanDA starts serves different providers of computing resources and from time to time in a specific way. On fist step we assumed, that all these differences will be covered by new pilot (Pilot 2.0). After analysing of already know workflows and new component model for this application we found that Pilot 2.0 will be too complicated for lightweight application, in terms of future support. More other, Pilots distribution machinery (aka Auto Pilot Factory) were reviewed, and was decided that improvements on this layer will be needed as well. That was the entry point for new PanDA application - 'Harvester'


MIRA
Edison
TITAN
Cloud A
Tier 2
Tier 1
Tier 2
Rucio: Distributed Data Management
Open Science Grid
NORDUGRID
Grid Solution for Wide Area Computing and Data Handling
Users
PanDA
Cloud B
Tier 2
Tier 1
Tier 2
Pilot factory
Condor
High Throughput Computing

Harvester: resource oriented application in PanDA

- Harvester is a resource-facing service between the PanDA server and execution environment for resource provisioning and workload shaping. It is a lightweight stateless service running on a VObox or an edge node of HPC facility to provide a uniform view for various resources.
  - For grid sites execution environment is pilot job
  - For HPC it can be special MPI application (Yoda in case of ATLAS)

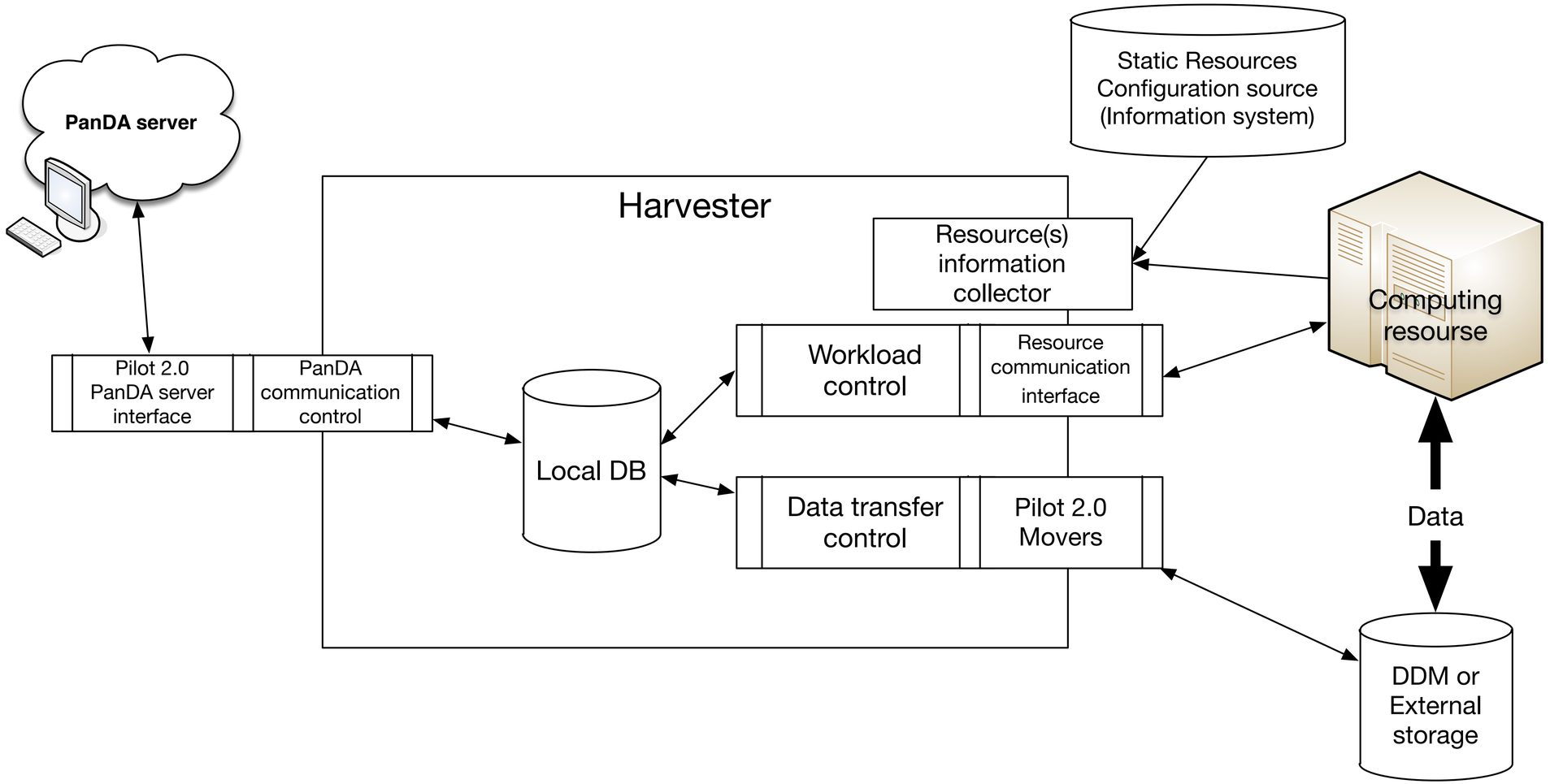
Harvester. Design key points (1)

- **Lightweigh**
  - To run on logon/edge nodes at HPC centers
- **Stateless for scalability + central database (oracle) + local database (sqlite3)**
  - Capability to rebuild the local database from the central database for auto restart
  - Local database to reduce redundant access to the central database
  - Only important checkpoints are propagated to the central database
- **Deployment with or without root privilege**

Harvester. Design key points (2)

- **Configurability**
  - To customize workflow for each type of resource
  - To turn on/off components with various plugins
- **Running on top of pilot API**
  - Core + plugins + resource specifics managers or pilot components
  - Leveraging development effort for the pilot consistently with the evolution plan (pilot 2.0)
- **Direct bi-directional communication with PanDA**
  - Requesting workload to PanDA based on dynamic resource availability information and static configuration
  - Receiving commands directly from PanDA to throttle or boost the number of workers (worker = pilot, MPI job, or VM)

Harvester. Design schema



Harvester. Current status and nearest plan

- Harvester core components and needful Pilot 2.0 components in active development stage
- Harvester core components were deployed and tested at OLCF.
  - Passed unit and functional test
  - Current deployment technics mostly oriented for developers: virtualenv as environment and components needful for main application control manager
- Rucio client was deployed as interface layer to transfer tools/protocols
  - Pilot 2.0 movers API development in progress
- In testing phase, SAGA based interface layer to manage payloads

Yoda: fine grained processing of ATLAS data on HPC

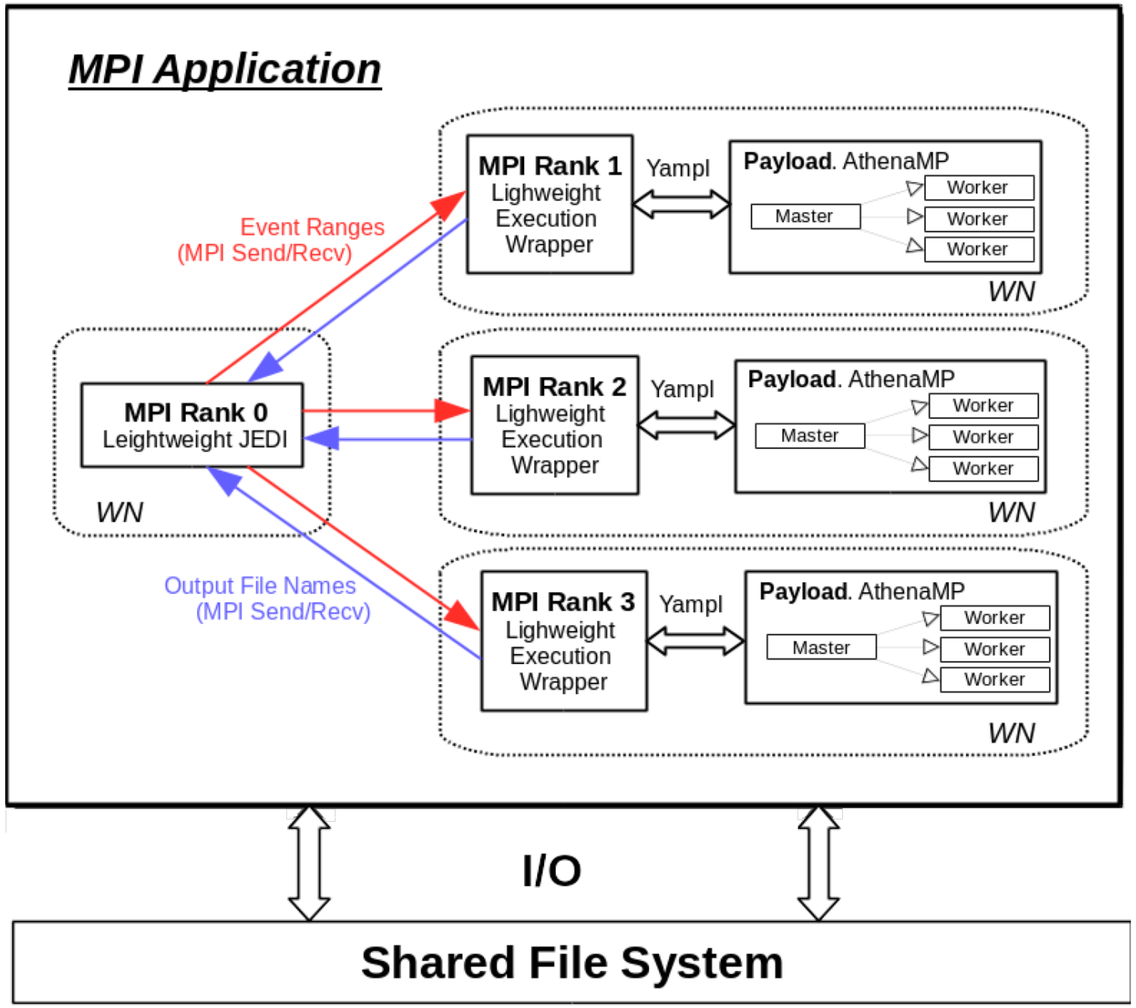
«The ATLAS Yoda system provides this capability to HEP-like event processing applications by implementing event-level processing in an MPI-based master-client model that integrates seamlessly with the more broadly scoped ATLAS Event Service. Fine grained, event level work assignments are intelligently dispatched to parallel workers to sustain full utilization on all cores, with outputs streamed off to destination object stores in near real time with similarly fine granularity, such that processing can proceed until termination with full utilization. The system offers the efficiency and scheduling flexibility of preemption without requiring the application actually support or employ check-pointing.»

21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) IOP Publishing
Journal of Physics: Conference Series 664 (2015) 092025

Yoda. Development stages

- Still in development stage
- Initially Yoda was designed as special extension of PanDA Pilot.
  - All complicity, related with asynchronous data transfers were integrated in to PanDA pilot
- With Harvester, Yoda was extracted from Pilot and adapted as standalone MPI application
  - Asynchronous data management naturally organised in Harvester

Yoda simplified schema



Yoda current status and plans for OLCF

- Yoda still on development stage
  - Early versions will be deployed at OLCF soon for testing
- Yoda on Titan will not use Object Store for stageout of produced data:
  - Projected amount of produced files is too big
  - Merging process in OLCF should be agreed (IO intensive operation)