PanDA@OLCF. Current solution.

There are two core applications used at OLCF to perform PanDA jobs.
'Launcher' - the application to spun and manage proper number of launched MultiJob Pilots
'MultiJob Pilot'- the application to perform set of PanDA jobs simultaneously like MPI/Parallel job.

'Launcher'
Daemon application, to support reasonable number of launched pilots on interactive node. Number of launched pilots defined by the number of available slots in queue of local job scheduler: to use resources in efficient way; and number of pilots which perform stag-out operation - to be able to throttle loading to node.
Also 'Launcher' perform collection of some statistic and some regular operation like proxy certificate and CRL updates.

'MultiJob pilot'
Application, based on PanDA pilot architecture and code base, which allow to launch set of PanDA jobs as one MPI task. Efficiently works to collect 'backfill' resources.

'MultiJob pilot' workflow
Basic workflow of MultiJob pilot is same as regular PanDA Pilot.
Here is major steps of regular Pilot:

- Get resource description (PanDA queue, DDM etc.)
- Get payload description (PanDA job)
- Start process 'Monitor' which monitor execution and interconnect with PanDA server (job updates, heartbeats etc.)
- Start process 'RunJob' which manage full execution process and send updates to 'Monitor' (through TCP)
- 'RunJob' perform:
    - Stage-in input data, using 'movers'
    - Prepare setup environment (vary for different Experiments)
    - Launch payload
    - Waiting till the end of execution of payload
    - Analyze results of execution and extract some additional information (vary for different Experiments)
    - Perform Stage-out (in case execution was successful)
- After finishing of payload (and RunJob)'
    - Performed final update
    - Archived and transferred logs

Here is list of modifications in MultiJob pilot
Retrieving of payload.
MultiJob pilot was instrumented with set of functions which collect dynamic information about available computing resources (number of available nodes, and duration of availability). According to this information pilot requests set of jobs from PanDA server in one call (getJob call was supported for bulk operation)
Preparing of sandboxes for each PanDA job in set
Each sandbox will contain input and output data for particular PanDA, will contain all other output of payload, will be used by payload like working directory
Updates of jobs info on PanDA server
Sequential operation through set of jobs. Update of one job ~1 sec. With big amounts of jobs may take significant time.

StageIn data
Sequential operation with using of pilot movers. GFAL mover used at OLCF. Operation optimized, by reducing of number of remote transfers in case file was already downloaded. Each job works with own copy of input file, to avoid high number of parallel reads.
Setup preparation
For optimization of launching of payloads, one time setup used for all bunch of PanDA jobs. Setup commands and environment variables declared in PBS script and will be used by all MPI ranks/paylods. It put a restriction for having ATLAS jobs with different releases in one submission.
Re checking of available resources and adjustment of number of jobs
Since, preparation of for launching may take significant time, rechecking of available resources needed before submission. In case pilot see less resources, number of jobs in this submission will be reduced. Jobs will be returned to PanDA server as 'closed'.
Payload preparation for MPI
To launch bunch of PanDA jobs as one MPI submission, simple MPI script was implemented. This script reads list of payloads (trfs) from input file, and launch each of them in one MPI rank.
Launching of execution
Pilot interconnect with local batch system to queue MPI job. Parameters of this MPI job optimaized by execution time and size with available unused resources. In most of cases, execution starts almost immediately. In case execution didn't started in five minutes, job will be canceled by Pilot and readjusted again.
All other operations: analyzing of results of execution, extraction of metadata from results, stage-out etc.   are sequential, but not highly time critical. During this operation pilot set special mark (file), that he is in stage-out mode and 'Launcher' may start next Multijob pilot.

ATLAS SW Deployment.
rel 19 - was deployed by using pacman
rel 21 - was deployed throw ayum. Also procedure of deployment by using of 'yumdownloader' was tested and give positive results.

Specific software deployment.
Only needful GRID component was deployed: literally GFAL with all dependencies and extensions, VOMS client.
RUCIO client installed as well.