Experience from a Pilot based system for ATLAS

Dr Paul Nilsson

University of Texas at Arlington
On behalf of the PanDA team

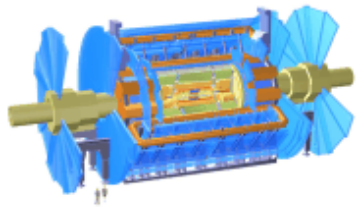
Outline

- **Panda introduction**
  - Features
- **Pilot** → Schedulers
  - Features
  - Job recovery
  - Workflow
  - Multitasking
- **Panda performance**
  - Efficiency
  - Error reporting
  - Central production
- **Condor glide-ins**
- **Conclusions**
  - More information

What is Panda?

167 - PanDA: Distributed production and distributed analysis system for ATLAS
Grid middleware and tools - Monday 03 September 2007 15:00
Presenter: MAENO, Tadashi (Brookhaven National Laboratory)

- PanDA is the ATLAS **Production** and **Distributed Analysis** system for OSG
- Developed by US ATLAS to meet the requirements for full scale production and distributed analysis processing for the ATLAS Experiment at CERN
- ATLAS places challenging requirements on throughput, scalability, robustness, operations manpower, efficient integrated data/processing management
- Current estimates are for 100-200k jobs/day within US ATLAS during full scale running, and ~4 times that number ATLAS-wide
- Panda development began in August 2005 and it took over US ATLAS production responsibilities in December 2005



Panda features

- Designed to support both managed **production** and **user analysis** via flexible job specification/injection
- **Dataset** based organization of Panda matches the DDM system and the **analysis work model** (based on ATLAS data management system DQ2)
- DDM used to **pre-stage input data** and **immediately return outputs**, all asynchronously, minimizes data transport latencies and delivers earliest possible results
- Management/optimization of workload via **job queue** with **late binding** of jobs to worker nodes gives **dynamic and flexible** system response to **highly variable DA work**
- Use of grid and farm batch queues to **pre-stage job wrappers** to worker nodes (**pilot jobs**) and **directly deliver workloads** from Panda allows **fast injection of DA work**

What does the Pilot do?

- Lightweight **execution environment** to prepare CE, request actual payload, execute payload, and clean up
- Handles data **stage-in** and **stage-out** between worker node disk and local SE through DDM client - data already copied to site by DDM system
- **Pilot jobs** broadcasted from Job Scheduler(s) to batch systems and grid sites; actual ATLAS job (payload) is scheduled when CPU becomes available, leading to low latency

Footnote about Schedulers

- Multiple pilot submission approaches
  - **Condor-G**, for US ATLAS production at most US production sites
  - **Local batch scheduler**. **PBS** at UTA, TRIUMF and UBC, **Condor** at BNL, AGLT2, TORONTO
    - Very efficient and robust, but cannot be centrally managed
  - **New generic scheduler** (‘AutoPilot’)
    - Supports local batch, Condor-G, LCG, and (soon) pilot factory (based on new Condor scheduler glide-in)
      - Extends Condor-G support OSG-wide and LCG-wide
  - **Dedicated submit hosts** at BNL, UTA, Madison, Lyon
  - **CERN host** for testing coming online

Pilot features

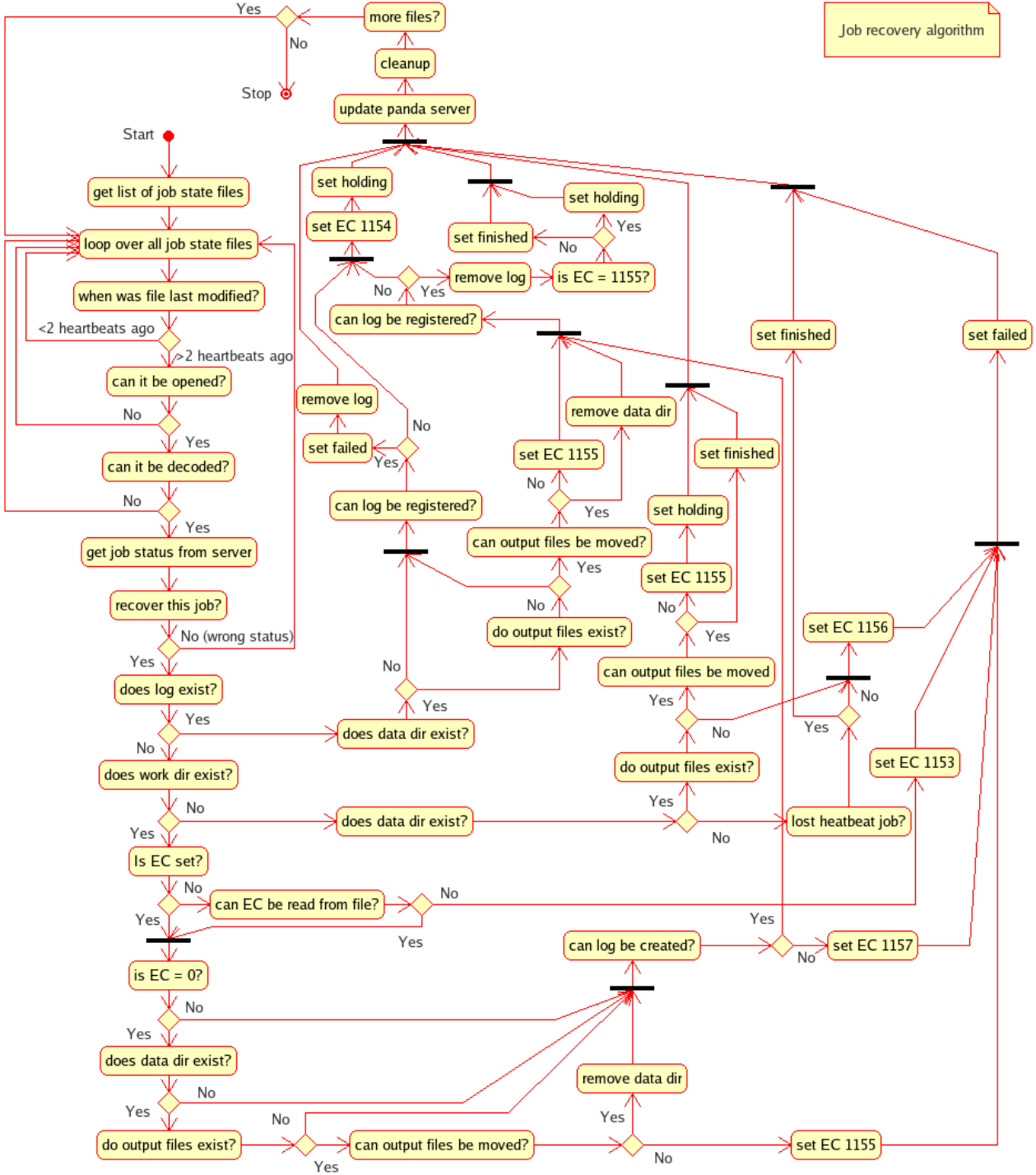
- **Data transfer** of input/output and log files to/from local SE
  - Pilots can be centrally configured according to each site policy
  - File transfers using various copy tools; *cp, dccp, rfcp, uberftp, gridftp, lcg-cp*
  - Direct reading support for dCache, Castor and Xrootd
  - Time-outs for [hanging] transfers (supports multiple attempts)
- **Job execution**
  - Pilot exits immediately if no real job is available
  - Pilot forks job wrapper for the real payload
  - Job wrapper communicates with pilot through TCP
  - Cleans up workdir after job is done (finished or failed) and saves the tarball of workdir to SE
- **Monitoring**
  - Job monitor running as separate thread - is job still alive? Kills sub processes if no output is produced within limit
  - Sends heartbeats to job dispatcher every 30 minutes
    - Can kill a job if desired by user (message sent through heartbeat communication channel)
  - Keeps current job state in a special file
- **Reports local DDM space** to panda monitor
- **Job recovery** – next slide

Job recovery

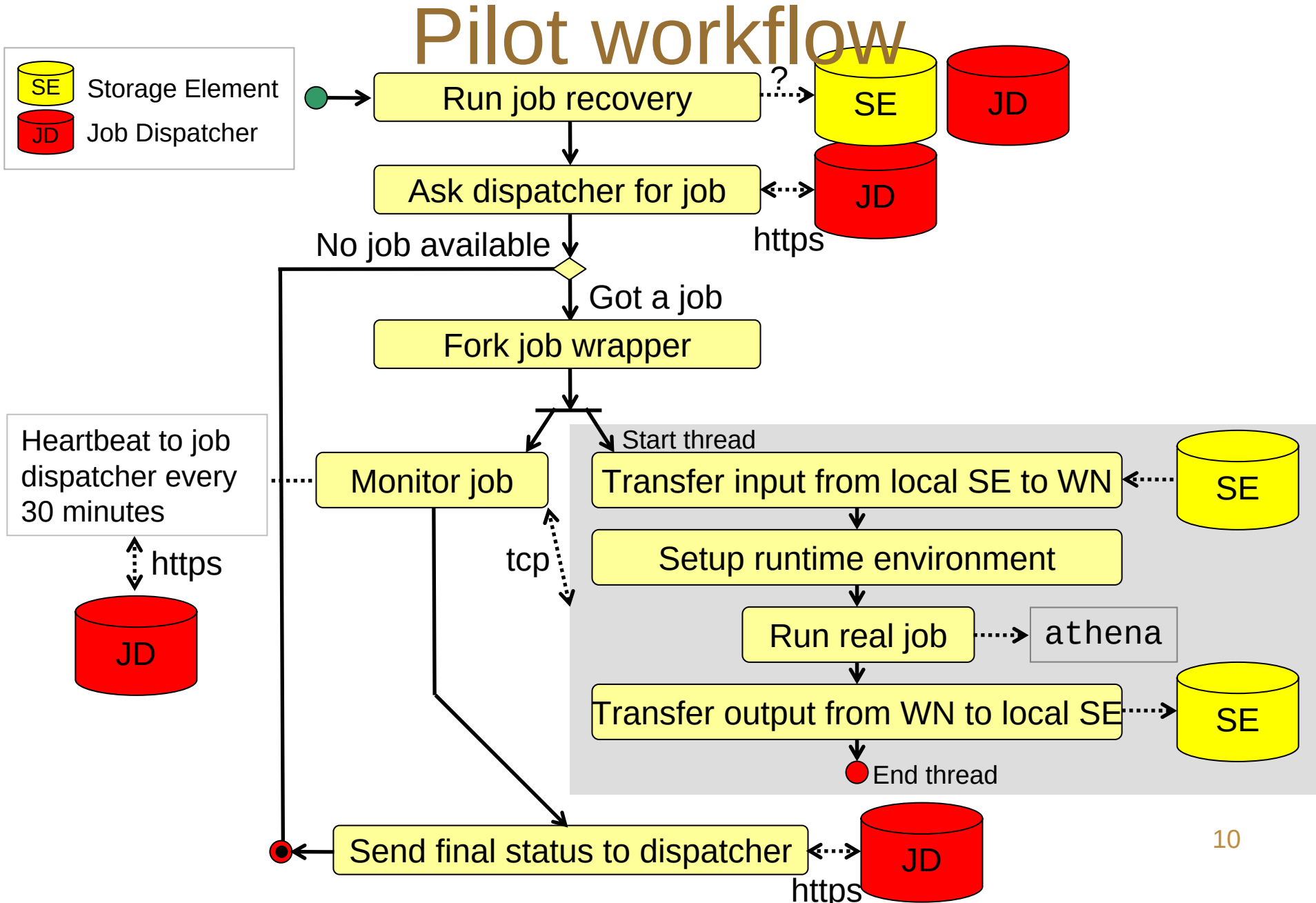
- When there are site or service problems, valid data from completed jobs can be stranded on WNs (or even deleted)
- Pilot is equipped with a recovery system to salvage these jobs, without a rerun
- If a completed job is unable to migrate its output to SE, it leaves job output and status on WN
- A later pilot checks for these stranded jobs, and retry the SE data movement
- Effective in waiting out service disruptions and recovering afterwards

  → CPU cycles otherwise lost are recovered
- Job recovery is optional

Job recovery algorithm

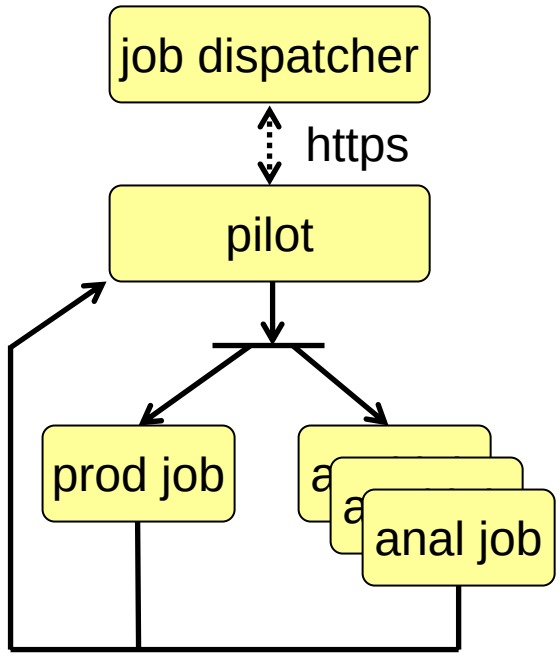


Pilot workflow



Multitasking Pilot vs queues

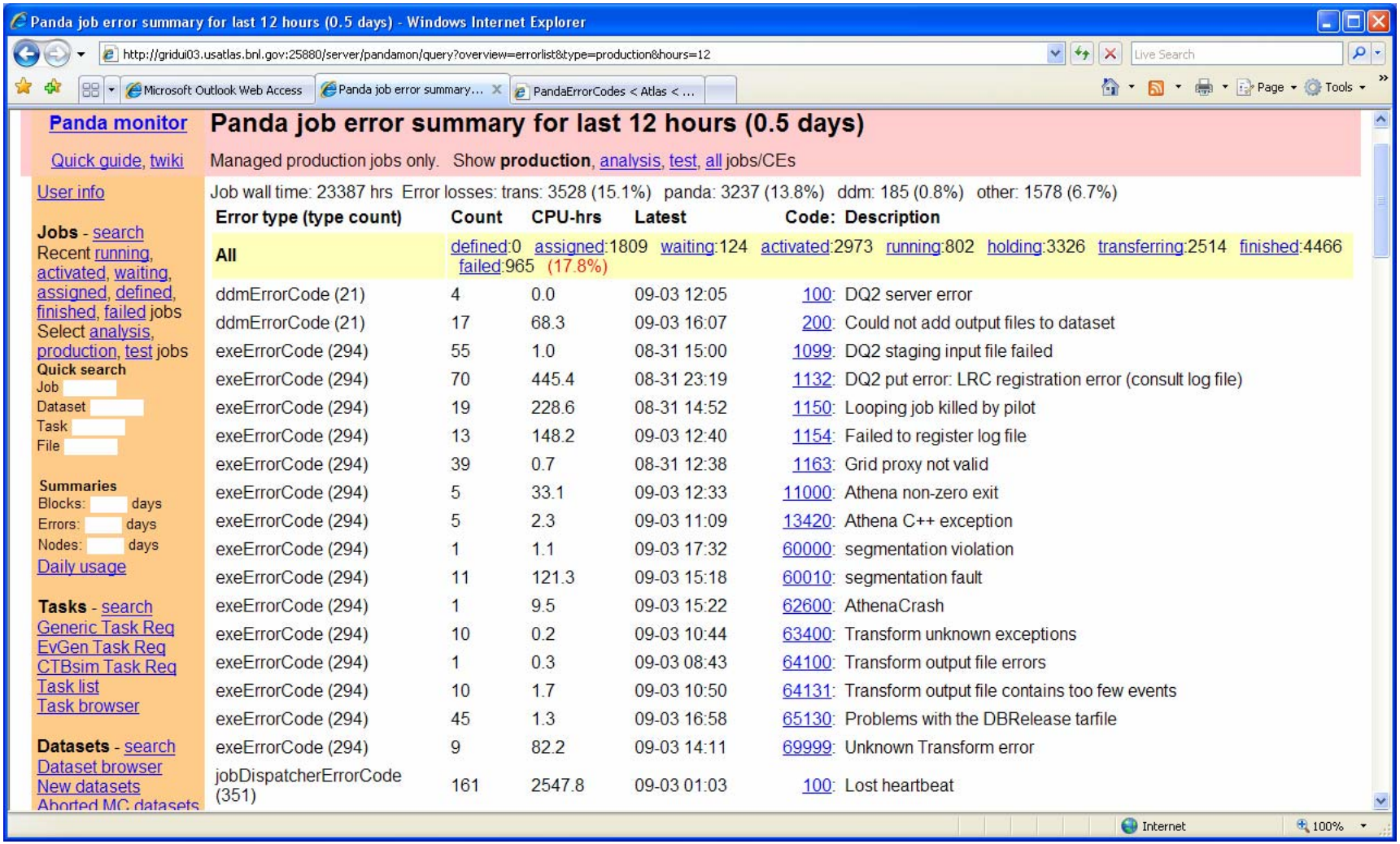
- Multitasking pilot – ability to run a production job at the same time as a user analysis job
  - Pilot asks for new analysis jobs one after another until production job finishes
- Can offer a large pool of analysis resources (the production farm) without machines standing idle
  - Good for accommodating highly fluctuating analysis loads; very short latency – always ready for DA jobs.
  - Concerns: Resource contention (esp. memory), conflict with resource usage policies, break “fair” sharing of local batch systems with other users’ jobs
- Pending evaluation, traditional approach of special queues (long and short) for analysis jobs is used



Panda efficiency

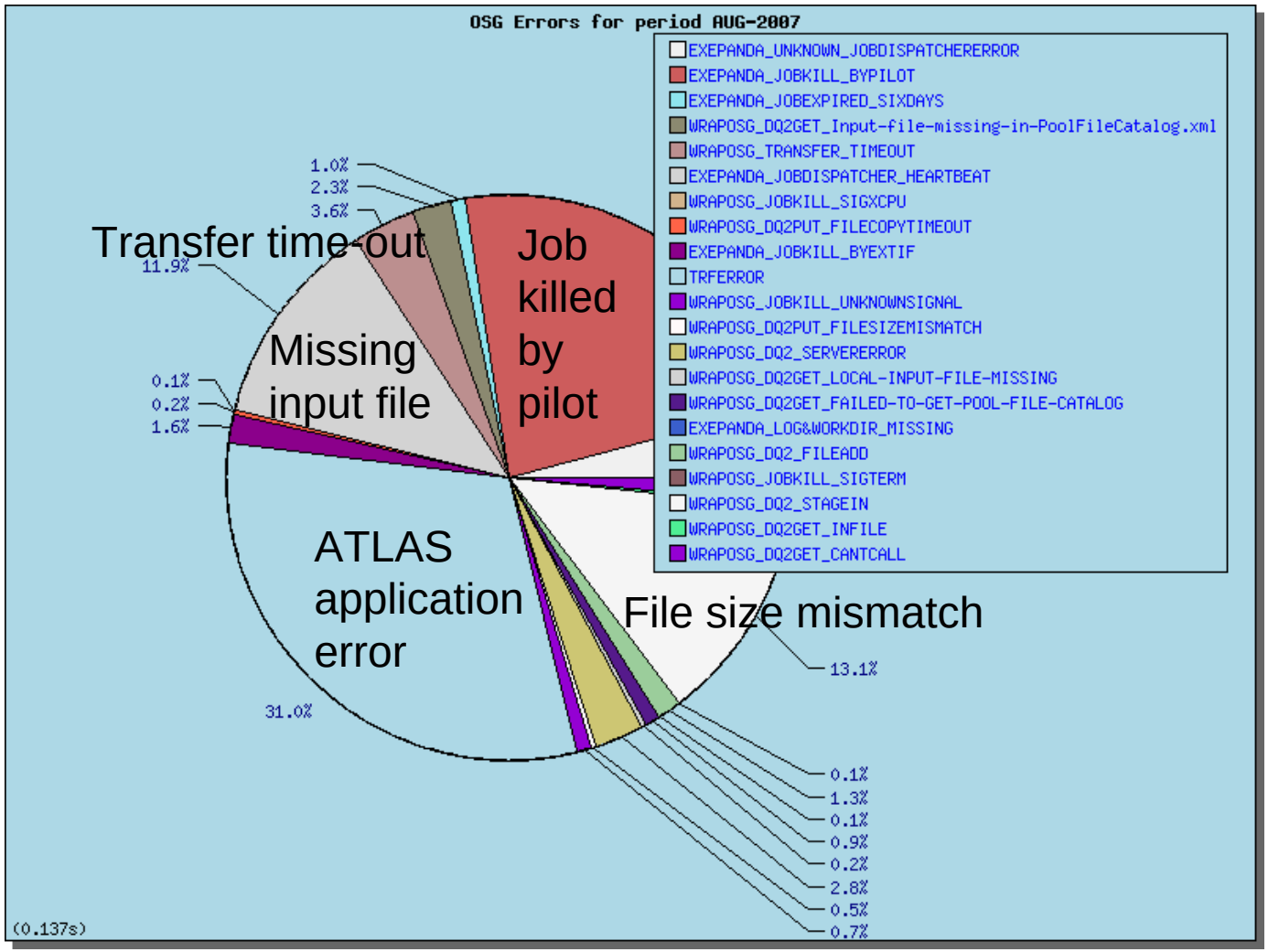
- Automated error analysis critical for scalability
  - Currently ~10k jobs/day – if 2% unknown failures, requires debugging of hundreds of log files/day, which is the most time consuming part of operations
  - Scaling to 100k+ jobs per day will be a challenge
- Logging, monitoring, and error reporting
  - Independent https based logging service
  - Integrated web based monitoring view (next slide)
  - Panda reports 60+ different errors (and growing with experience)
- Panda system errors so far < 3% (majority of errors are site problems, or from other software systems)
  - Found Panda system errors usually fixed immediately
- No Panda scaling problems seen so far, with a maximum of ~40k jobs in job queue

Error reporting (1/2)

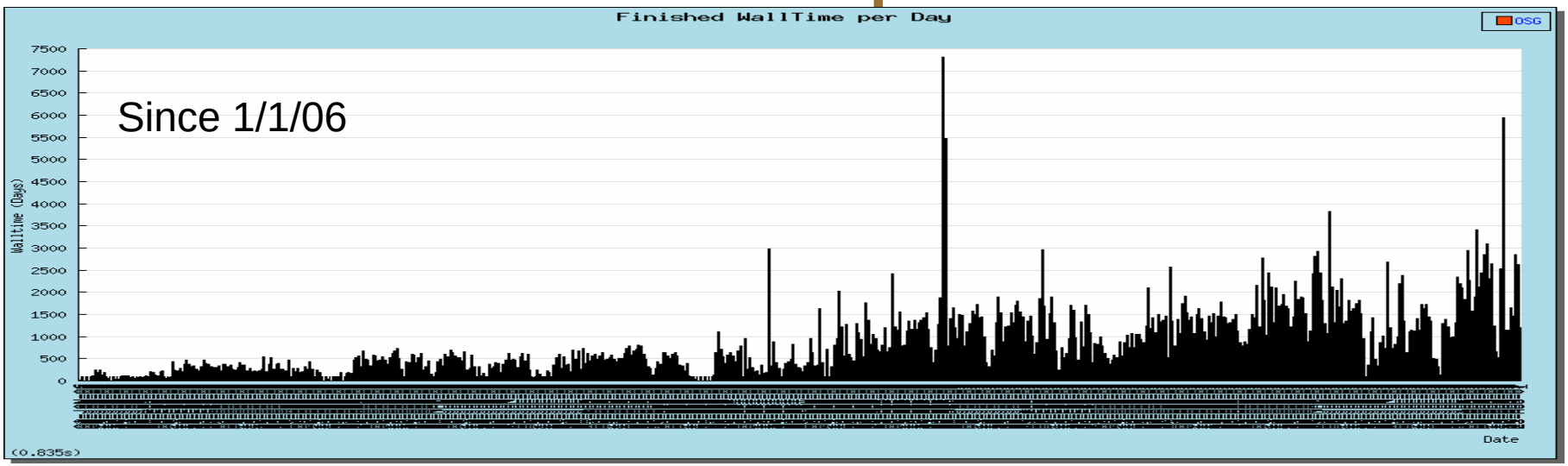


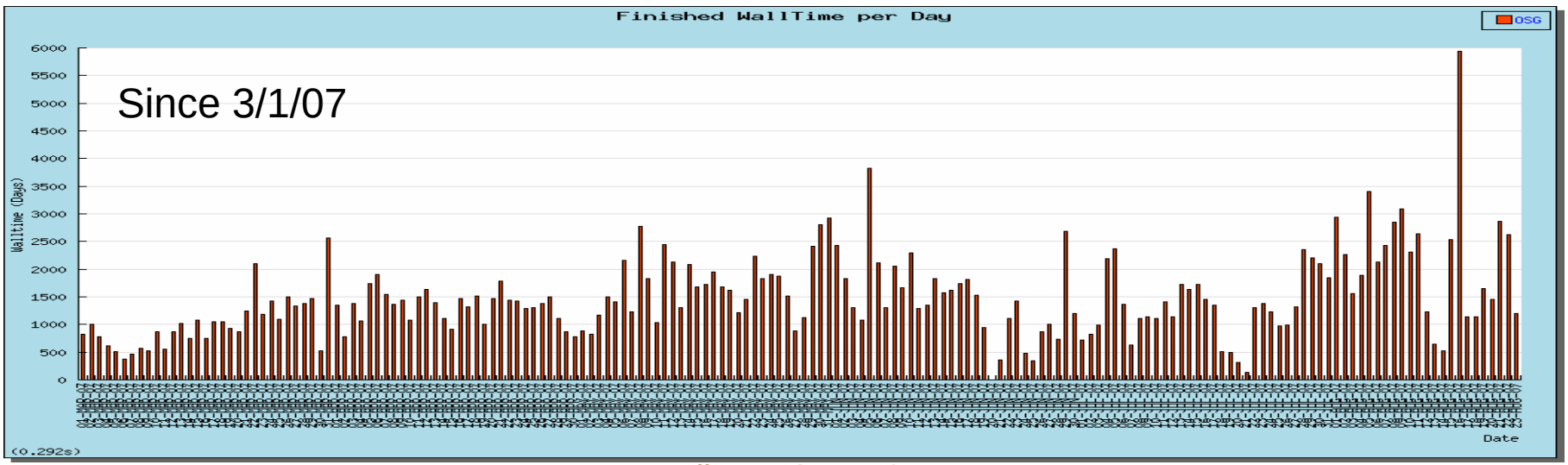
Panda monitor

Error reporting (2/2)

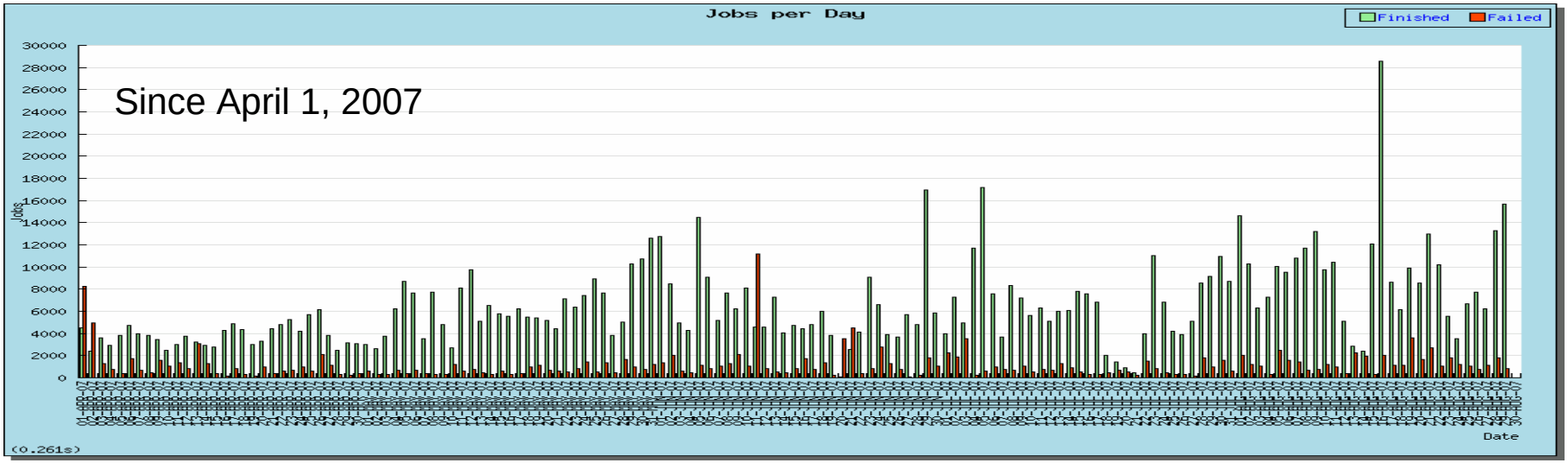


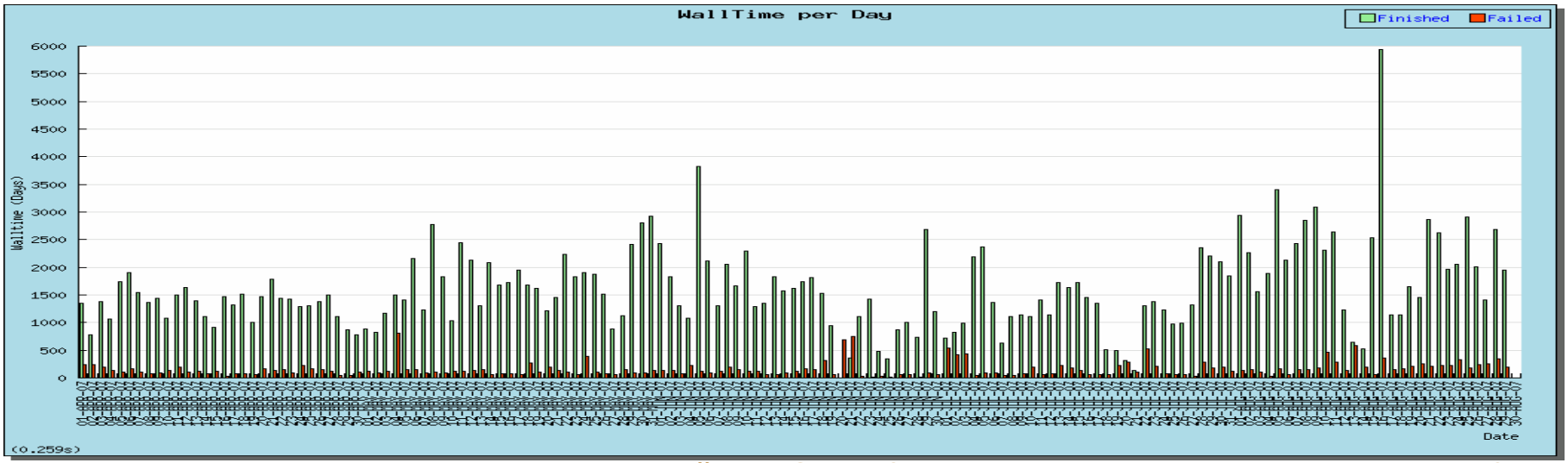
Panda central production





Job and walltime efficiencies





Panda and Condor Glide-ins

- In the plans since October 2005, actively pursued since September 2006
  - Initial target is a new capability for Condor: schedd glide-ins to support site-level pilot factory (and thus move away pilot submission from a global submission point as we have now)
  - Condor only supports startd (‘pilot’ type) glide-ins at present
  - Working directly with Condor team
  - In OSG we are collaborating with CMS (Igor Sfligoi, FNAL) on startd glide-in
- Panda can easily use Condor startd glide-in pools to submit jobs

Conclusions

- Panda production on OSG is going very well (CPU's busy, no scaling limits or performance, latency issues found)
- Pilot equipped with highly useful features; esp. job recovery and multitasking
- Many different storage systems supported
- Several different pilot delivery systems
- Panda with Condor glide-ins under development

More information

- Panda home page https://twiki.cern.ch/twiki/bin/view/Atlas/Panda
- Panda pilot home page https://twiki.cern.ch/twiki/bin/view/Atlas/PandaPilot
- Panda monitor http://services.atlascomp.org/?redirect=pandamon
- Open Science Grid http://www.opensciencegrid.org/
- ATLAS Experiment http://atlas.web.cern.ch/Atlas/index.html