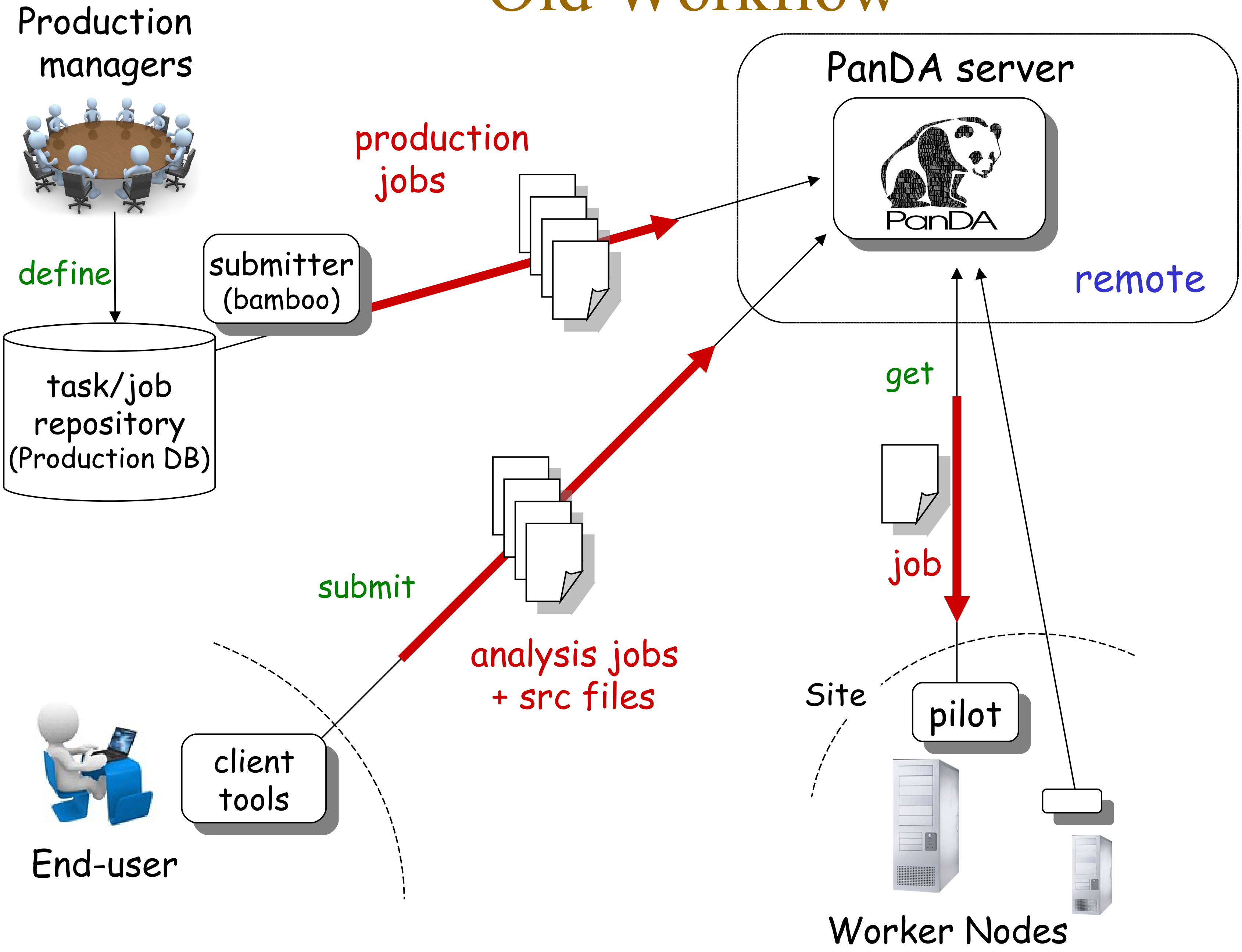
New Features in PanDA

Tadashi Maeno (BNL)

Old Workflow



Production managers

define

task/job repository (Production DB)

submitter (bamboo)

production jobs

PanDA server

remote

End-user

client tools

submit

analysis jobs + src files

get

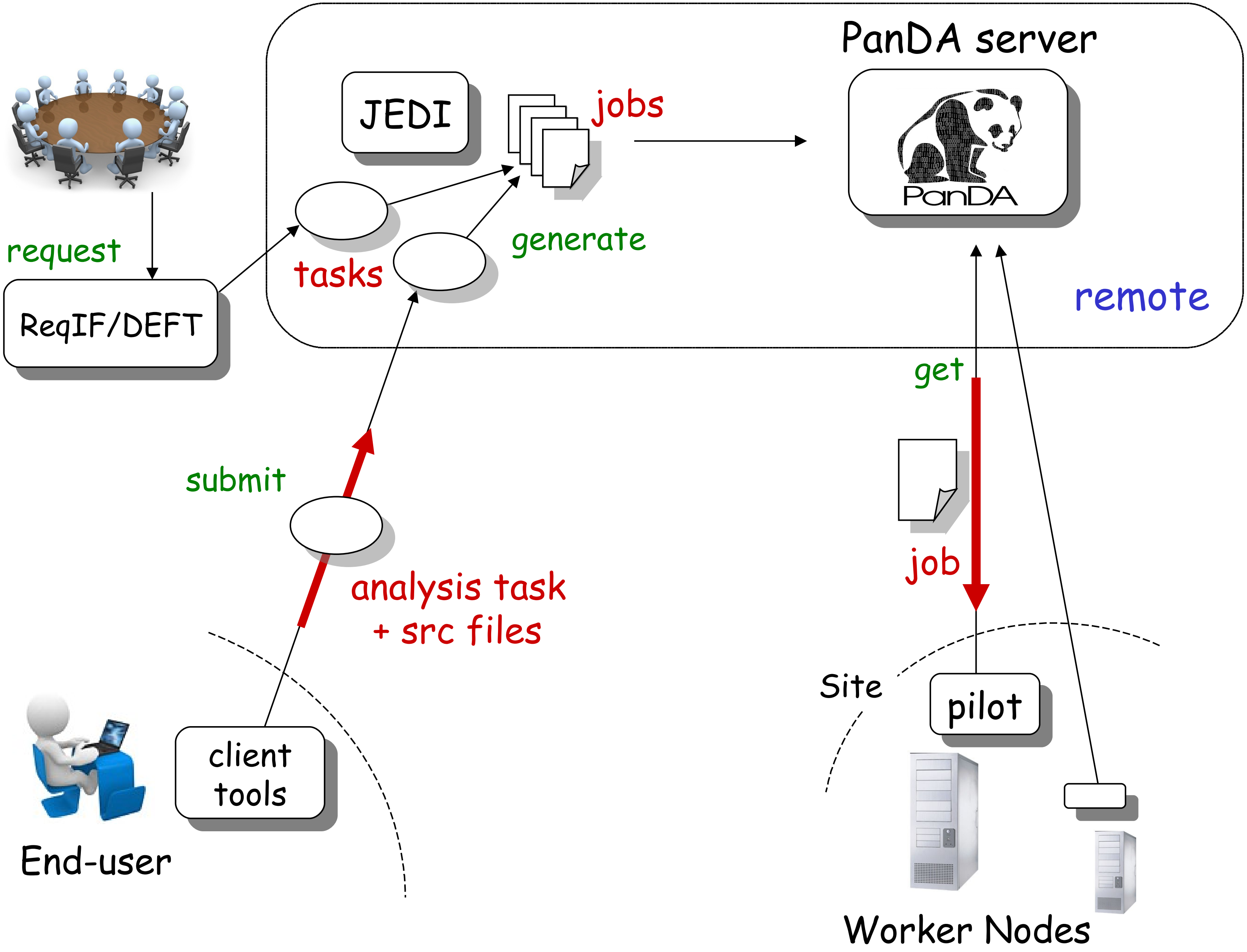
job

Site

pilot

Worker Nodes

New Workflow



Main Changes

- Tasks are submitted to the system instead of jobs
  - Task = runs on input datasets to produce output datasets
  - Job = runs on input files to produce output files
- Many client functions (job submission, retry, kill) are moved to the server-side
- Built-in retry mechanism at the same site and/or other sites
- Capability for task chaining
- Optimization based on job profile measured by scouts
- Tasks are more exposed to users rather than jobs

Benefits

- The system works more coherently with user's perspective
  - Users are interested in tasks rather than jobs
  - Allows task-level workload management
    - Retry/kill/reassign tasks
- Simplification of client tools and centralization of user functions
  - Better maintainability
- Optimal usage of computing resources without detailed knowledge on the grid
  - Lower hurdle for users
- Optimization of database access to get/provide task information

Main functions

- Scout jobs
- Dynamic job generation
- Automatic retry
- Task Chaining
- Prestaging from TAPE
- Merging
- Network-aware brokerage

Scout jobs

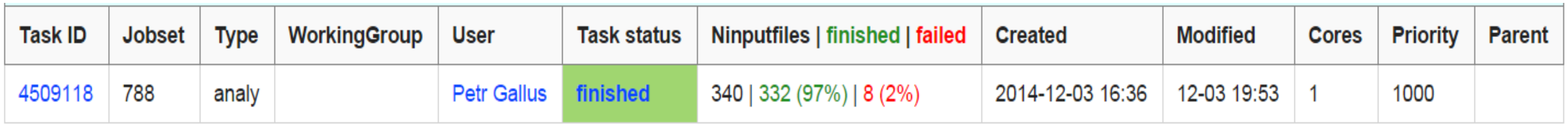
- Scout jobs are introduced to measure job profiles before generating a bunch of jobs
  - Scout-avalanche chain
- If no scout jobs succeeded the task is aborted
  - The user can avoid filling up the system with problematic jobs
- Job parameters are optimized by using real job profiles, which is more accurate than a-priori estimate by users
  - Some users unintentionally submit short jobs to long queues since they don't know beforehand how exactly long their jobs take

Dynamic job generation

- Input for jobs is optimized for each site/queue
  - e.g., more input files for larger scratch disk and/or longer walltime limit
- Considering # of cores and memory size as well
  - Good for MCORE and exotic resources
- The number of input files for each job could vary even in the same task
  - Tasks can be configure for all jobs to have the same number of input files if necessary

Automatic retry

- JEDI has a capability for automatic retry at the same or another site
- Input is dynamically split or combined based on site/queue configuration
  - For example, if jobs are reassigned to a site where longer walltime limit is available, jobs are reconfigured to run on more input files
- Jobs are not atomic any more
  - PandaMon shows ratio between successful and failed inputs and hides retried jobs



Task ID	Jobset	Type	WorkingGroup	User	Task status	Ninputfiles | finished | failed	Created	Modified	Cores	Priority	Parent
4509118	788	analy		Petr Gallus	finished	340 | 332 (97%) | 8 (2%)	2014-12-03 16:36	12-03 19:53	1	1000	

Task Chaining

- A capability of task chaining is available
- Example
  - Completion of an evgen task can trigger a G4 simulation task
- The downstream task starts processing before or when the upstream is completed, according to task definition
  - e.g., G4 jobs can be generated as soon as new EVNT files become available

Prestaging from TAPE

- When inputs are available on TAPE, subscriptions/rules are made in DDM and jobs are activated once callbacks are received
  - The same machinery has been used for production and now is used for analysis as well
- More TAPE usage
- Will use separate shares for production and analysis in the future so that those activities would not interfere with each other

Merging

- JEDI has a capability to merge files in a task
  - To merge files at T2 before transferring back to T1
  - To merge user's output files
- Jobs go to merging state
  - Files actually start being merged when all jobs at the site finished/failed
- Possible to have a separate task to merge files
  - e.g., super merge and log merge

Network-aware brokerage

- JEDI brokerage is aware of network performance measured by cost matrix
- Currently only for analysis
- Example
  - When siteA has free CPUs and good network connection to siteB where input data is available, jobs can be sent to siteA even if siteA doesn't have the input data. The jobs run at siteA and remotely read data from siteB
  - Jobs can go to siteA and siteB while jobs went only to siteB in the old system
    - Users can use more CPU resources in the new system

Future Plans

- To improve the brokerage to take the site failure rate into account
  - e.g. avoid problematic sites for reattempts
- To use pre-merged files as final products when merge jobs fail
  - Currently pre-merged files are discarded when merge jobs fail
  - For analysis