



Ask PanDA Project

MCP Agents for PanDA

Paul Nilsson (BNL)

July 30, 2025

RAG4EIC



Introduction

- “Ask PanDA” is an R&D project within the REDWOOD project

REDWOOD is a 5-year project funded by DOE ASCR, started in 2023

Lead PI: A. Klimentov (BNL)

- “*Resilient Federated Workflows in a Heterogeneous Computing Environment*”
- **Collaborating Institutions:** Brookhaven National Laboratory (BNL), Oak Ridge National Laboratory (ORNL), SLAC National Accelerator Laboratory, University of Pittsburgh, University of Massachusetts Amherst, Carnegie Mellon University

- A main objective of REDWOOD is the optimization of WFMS (WorkFlow Management System) to enhance system resilience
- The idea behind “Ask PanDA” is to (via PanDA Monitor) make it easier for users and operators to
 - Learn about the PanDA system (all related tools and components)
 - Understand task and job failures (def. *task*: *collection of grid jobs using the same dataset*)
- Related efforts include improved error descriptions and categorizations for grid job failures
- **Overall goal:** Improved system resilience by reducing number of failed jobs
 - (not only by having better informed users and operators, but also actively by optimizing number of retries of failed jobs – an outcome of proper error categorization)

Ask PanDA Service

Purpose & Evolution

- Originally built to improve access to PanDA documentation
- Now supports both **static** and **dynamic** question answering
- Rewritten using **Model Context Protocol (MCP)** for modularity

MCP Advantages

- Supports agentic workflows
- Hides backend complexities
- Enables plug-and-play LLM compatibility

LLM Integration

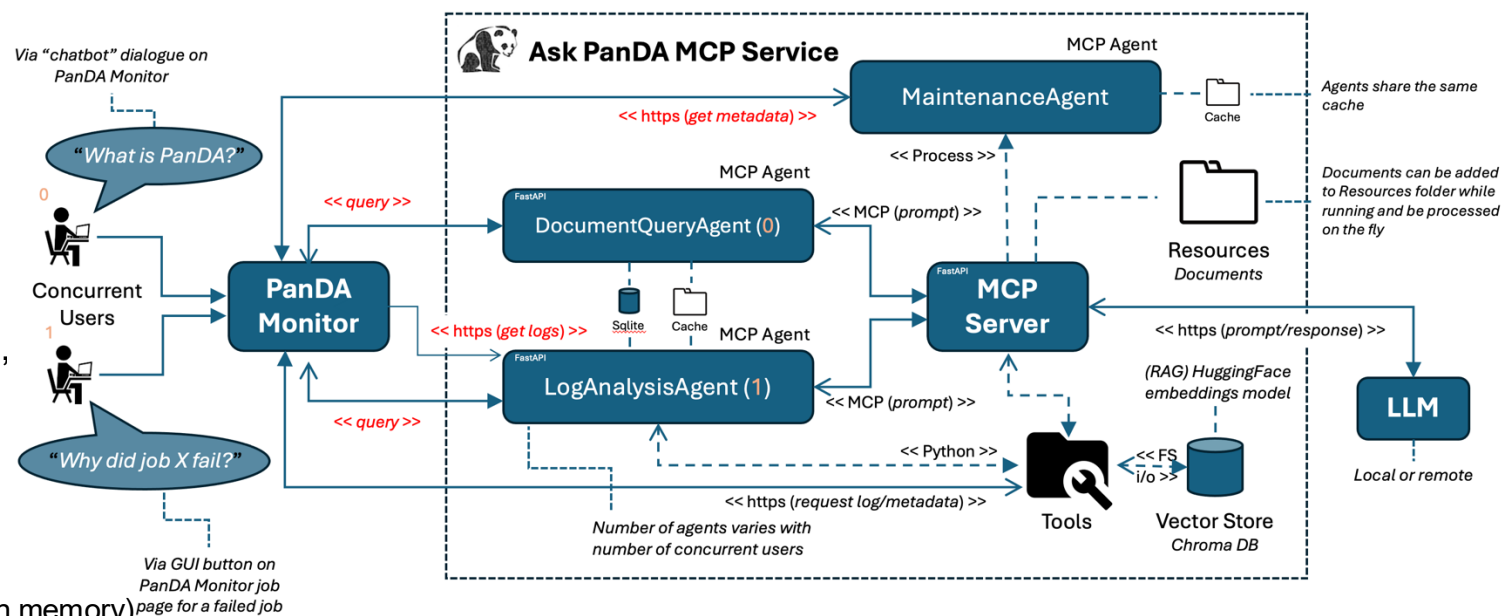
- **Via API:** OpenAI (GPT), Gemini, Anthropic (Claude), LLaMA
- Demonstrated special version for Claude Desktop

Architecture Overview

- Composed of a **server** and multiple **agents**
 - **Document Query Agent:** general chatbot (with memory)
 - **Log Analysis Agent:** explains job failures
 - **Data Query Agent**^{*NEW*}: analyzes special JSON files
 - **Selection Agent**^{*NEW*}: common interface for LLM agents
 - **Maintenance Agent:** supplies metadata to others, cleans up old conversations, etc

Question Types

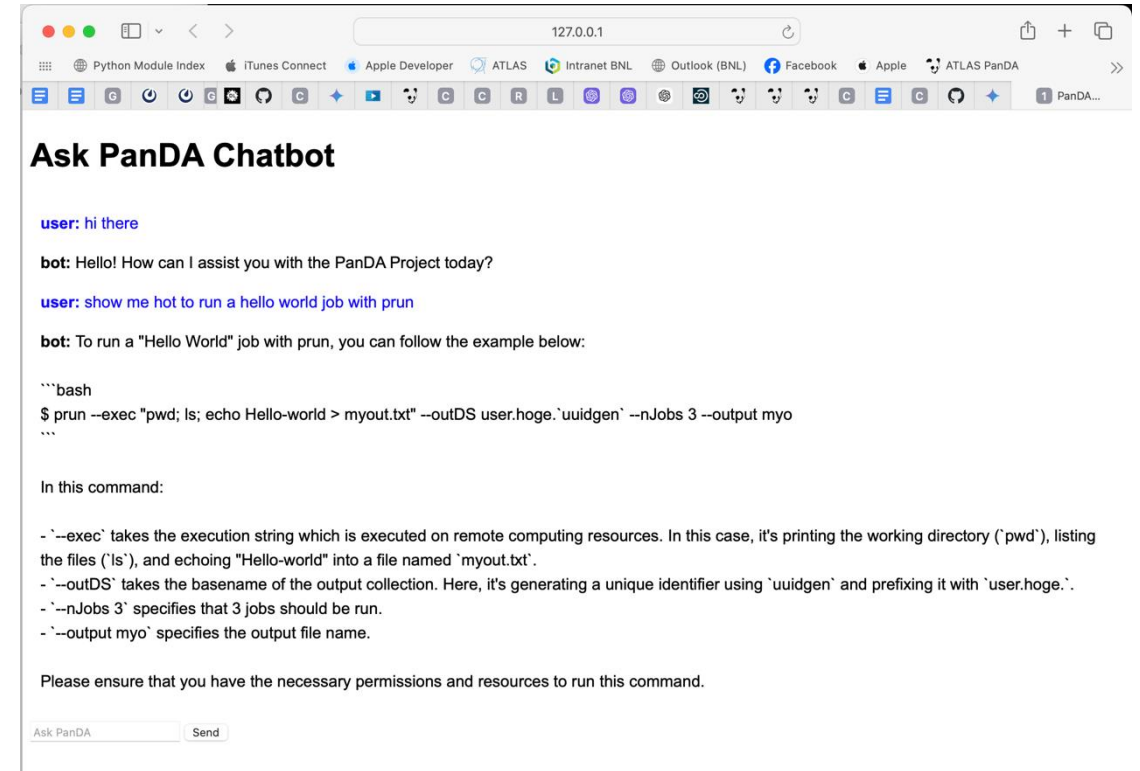
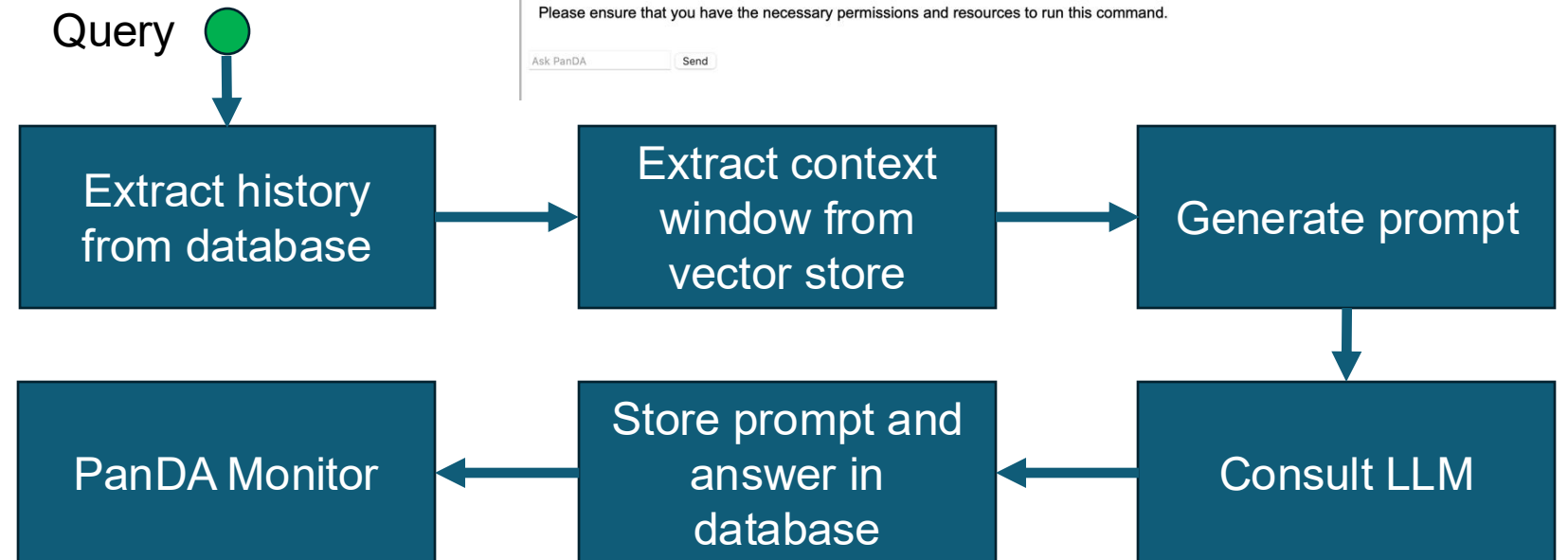
- **Static** (direct LLM queries): e.g., *"What is PanDA?"*
- **Dynamic** (requires interpretation – via LLM): e.g., *"Why did job XXX fail?"*, *"What's going on with task YYY?"*
- (Selection Agent handles both)



- Relevant server functions are asynchronous to achieve concurrency
- Server rebuilds vector store upon start – automatic updates for new documents in resources folder
- GitHub repository: <https://github.com/BNLNPPS/ask-panda>

Document Query

- A functioning “chatbot” was demonstrated early this year →
- Current functionality implemented as MCP agent (“Document Query Agent”) for future integration into proper chatbot in next generation PanDA Monitor
- Context stored in ChromaDB vector store
 - HuggingFace and LangChain OpenAI models used
 - Contains PanDA and Rucio documentation
- Conversation memory stored in SqliteDB
 - Shared between agents (in progress)
- For testing purposes sufficient to use free Gemini LLM model



Document Query Example (Gemini)

Free Gemini API + HuggingFace all-MiniLM-L6-v2 (free embeddings model)

```
$ python3 -m agents.document_query_agent --question="What is PanDA?" --model=gemini --session-id=123
```

```
..
```

```
2025-07-22 23:03:57,220 INFO:Answer:
```

```
{'session_id': '123', 'question': 'What is PanDA?', 'model': 'gemini', 'answer': 'PanDA provides advanced Web based monitoring for different groups of PanDA users: scientists, developers, operators, managers, shifters. Panda monitor can also serve a data source for users scripts and custom automatizing.'}
```

```
$ python3 -m agents.document_query_agent --question="I don't think that was a good explanation, please explain it better" --model=gemini --session-id=123
```

```
..
```

```
$ 2025-07-22 23:06:24,072 INFO:Answer:
```

```
{'session_id': '123', 'question': "I don't think that was a good explanation, please explain it better", 'model': 'gemini', 'answer': 'PanDA is a workload management system (WMS) used for distributed computing, particularly in high-energy physics. It helps manage and monitor tasks and jobs submitted to a distributed computing environment, ensuring efficient resource utilization and providing tools for scientists, developers, and operators to track the progress and status of their work. It offers web-based monitoring and serves as a data source for custom automation.'}
```

Document Query Example (OpenAI)

OpenAI API + HuggingFace all-MiniLM-L6-v2 (free embeddings model)

```
$ python3 -m agents.document_query_agent --question="What is PanDA?" --model=openai --session-id=124
..
2025-07-22 23:12:34,431 INFO:Answer:
{'session_id': '124', 'question': 'What is PanDA?', 'model': 'openai', 'answer': 'PanDA stands for Production and Distributed Analysis. It is an API object that provides advanced web-based monitoring for different groups of PanDA users, such as scientists, developers, operators, managers, and shifters. PanDA monitor can also serve as a data source for user scripts and custom automatizing. It provides basic information needed to monitor jobs and tasks submitted into PanDA, with two views for navigation over tasks: Tasks list and a Task view.'}
```

OpenAI API + OpenAIEmbeddings (commercial model via LangChain)

```
$ python3 -m agents.document_query_agent --question="What is PanDA?" --model=openai --session-id=125
..
2025-07-22 23:24:59,168 INFO:Answer:
{'session_id': '124', 'question': 'What is PanDA?', 'model': 'openai', 'answer': 'PanDA, or Production and Distributed Analysis, is a data-driven workload management system designed to operate at massive data processing scales. It is capable of adapting to emerging computing technologies in processing, storage, networking, and distributed computing middleware. PanDA was developed to meet the production and analysis requirements of ATLAS and has demonstrated scalability over the last decade. The system allows for the monitoring of tasks and jobs, providing different views for users and developers.'}
```

Document Query Example (Gemini+OpenAI)

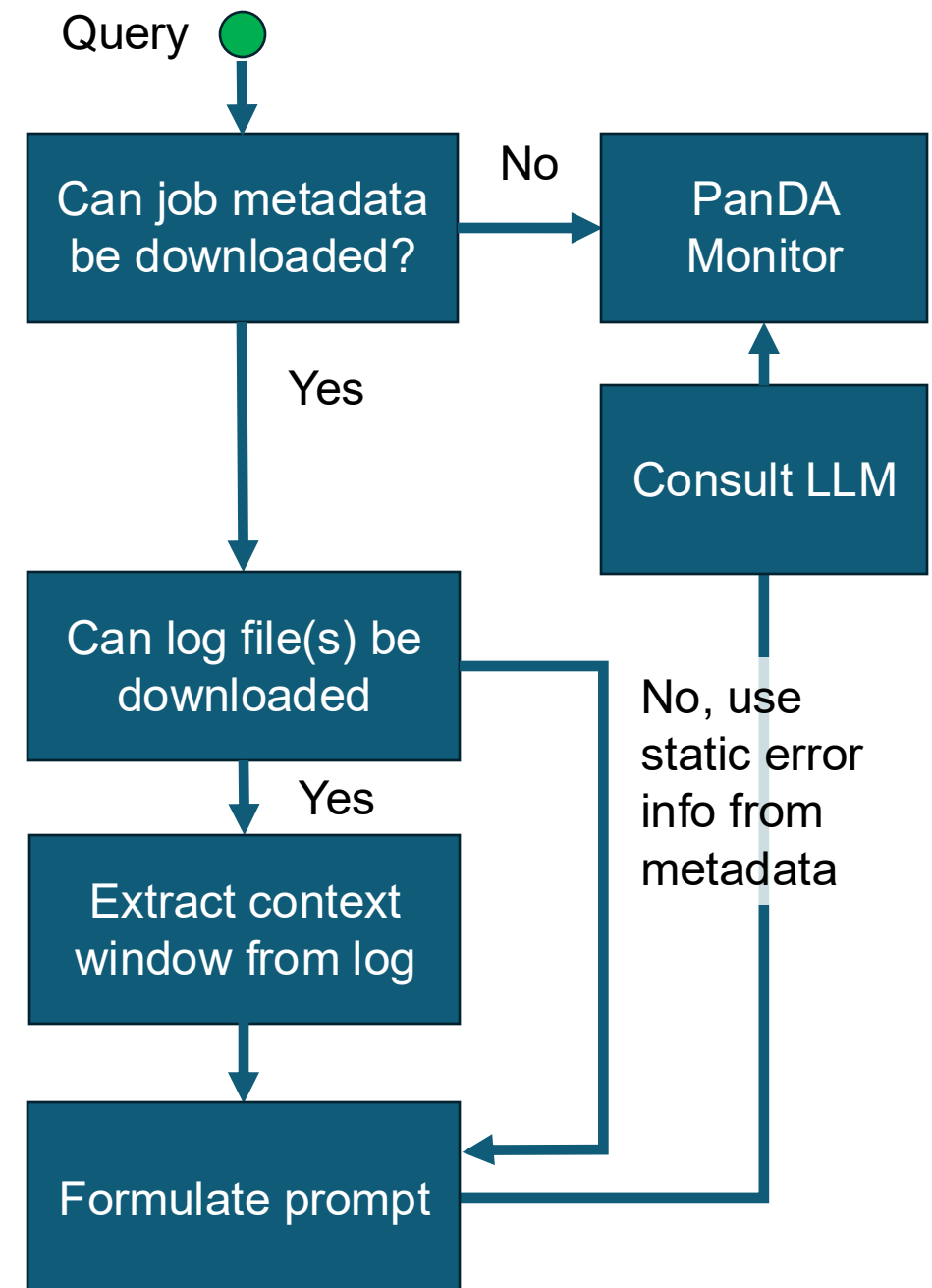
Free Gemini + OpenAIEmbeddings (commercial LangChain model)

```
$ python3 -m agents.document_query_agent --question="What is PanDA?" --model=gemini --session-id=126
..
2025-07-22 23:30:07,579 INFO:Answer:
{'session_id': '126', 'question': 'What is PanDA?', 'model': 'gemini', 'answer': 'The Production and Distributed Analysis (PanDA) is a data-driven workload management system capable of operating at massive data processing scale, designed to have the flexibility to adapt to emerging computing technologies in processing, storage, networking and distributed computing middleware. PanDA has been developed to meet ATLAS production and analysis requirements for a data-driven workload management system capable of operating at the LHC data processing scale.'}
```

→ Not only the API plays a big role in the quality of the answer, but also the embeddings model

Log Analysis

- For questions like “*why did my job fail?*”, LLMs will of course not know anything unless prompt is carefully prepared
- “Log Analysis Agent” performs multiple steps before asking LLM what went wrong
 - Script first downloads job metadata to learn general details about job (esp. state, error info)
 - Relevant log will be downloaded from PanDA
 - Extract context window (log messages immediately prior to failure)
 - Construct detailed prompt (enhanced with known error diagnostics to aid analysis) and send to LLM
- Response is returned from model (dictionary) and presented to user *[currently only CLI]*
- *TODO: store generated query and dialogue in memory*
 - Allows for dialogue to continue if there are questions about given explanation



Enhancing the Prompt

- An LLM will provide a better answer if it is guided towards an explanation
- In the case of log analysis, the agent uses known error explanations
 - It is better if those explanations are as detailed as possible (not just “stage-out failed” e.g.)
- In a recent effort, error codes + known [limited] descriptions were collected from all components

PanDA: Server error codes

- Supervisor: 95
- Task buffer: 28
- Data service: 8
- Job dispatcher: 6
- Executor: 3
- **Total: 140**

PanDA: WNScripts error codes

- runAthena: 10
- runHPO: 8
- runMerge: 8
- runGen: 7
- buildGen: 5
- buildJob: 4
- preGoodRunList: 3
- **Total: 45**

PanDA: Pilot error codes

- **Total: 145**

Transform (payload) error codes

- **Total: 70 (ATLAS specific)**

Others

- Other tools and scripts
- Needs special treatment (currently leads to generic error code seen by pilot)
- **Total: ~30**

Total all components: 430

Improved and explained error messages

What Do All Those 430 Errors Mean?

- All source codes have been analyzed by ChatGPT to provide better/longer explanations
 - Being reviewed by experts, but explanations appear reasonable – good enough as a start
- A new DB table was created (and populated) with extended error diagnostics
- The more detailed explanations are displayed on failed job pages in PanDA Monitor
 - Also added to prompt as a starting point/hint for LLM analysis to explain errors in logs**

Job error summary:			
Component	Code	Message	Description
Pilot error	1150	Looping job killed by pilot	The LOOPINGJOB error is used to detect and handle payloads that appear to be 'looping' ı meaning they are running without producing any output for an extended time. This is a typical symptom of jobs that are stuck in an infinite loop or have stalled in a way that is not immediately fatal but effectively unproductive.

Transform errors

Error code	Detailed explanation
1	The NEEDCHECK error means that the job requires manual intervention. This typically indicates that something went wrong during the job's execution, and it cannot proceed without someone checking and resolving the issue. It is a signal to the user or operator that they need to look into the job's details, logs, or configuration to understand and fix the problem before the job can continue or be rerun.
3	The TRF_SETUP exit code is used to indicate a "Transform setup error". This means that there was an issue during the initial setup phase of the transformation process. This could involve problems with configuration files, environment settings, or other prerequisites that need to be in place before the transformation can proceed.
4	The TRF_ARG_CONV_FAIL error represents a failure to convert transform arguments to the correct type. This error occurs when the transformation process encounters an issue while trying to convert the provided arguments into the expected data types.
5	The TRF_ARG_OUT_OF_RANGE error is an exit code used to indicate that an argument provided to a transform is outside the allowed range of values. This error helps identify when an input value is not within the expected limits, which can prevent the transform from executing correctly.
7	The TRF_ARG_ERROR error indicates that there is an issue with one of the arguments provided to the transform. This could be due to various reasons such as: The argument is not recognized by the transform. The argument is in an incorrect format. The argument is missing required components. The argument does not meet the expected criteria or constraints.
8	The TRF_ARG_DATASET exit code represents an error that occurs when there is an inconsistent dataset value in the input file list. This means that the dataset values provided in the input files do not match the expected or required format, causing the transform process to fail.
9	The TRF_INPUT_FILE_ERROR error is an exit code used to indicate that there is a problem with one of the input files that the transformation process is trying to use. This could be due to various reasons such as the file being missing, corrupted, or not in the expected format. When this error is encountered, it means that the transformation process cannot proceed correctly due to issues with the input file.
11	The TRF_OUTPUT_FILE_ERROR error indicates that there was a problem when handling a transform output file. This could mean that the program encountered an issue while trying to create, write to, or manage an output file during its execution.
12	The TRF_GRAPH_ERROR error code indicates a problem in setting up the substep graph. This error occurs when there is an issue with configuring the sequence of steps (substeps) that the transformation process needs to follow. The substep graph is essentially a representation of the workflow or the sequence of operations that need to be executed. If there is a problem in defining or connecting these steps correctly, the TRF_GRAPH_ERROR is raised.
13	The TRF_EXECUTION_PATH_ERROR error indicates that there is a problem trying to generate the requested outputs from the given inputs because the graph does not provide a path to do this. In simpler terms, it means that the system cannot figure out how to process the input data to produce the desired output, likely due to a configuration or setup issue in the processing steps.
14	The TRF_KEY_ERROR error represents a key error in a transform's internal dictionary. This error occurs when the code tries to access a key in a dictionary that does not exist. In Python, dictionaries are collections of key-value pairs, and attempting to access a non-existent

Task buffer errors

Error Code	Explanation
100	The EC_Kill error means that a task or job was terminated or killed. This could happen for various reasons, such as manual intervention by an administrator, a system decision to stop the task, or other conditions that necessitate stopping the task before it completes naturally.
101	The EC_Transfer error typically indicates a transfer timeout. This means that a data transfer operation has taken too long to complete and has timed out. This could happen due to network issues, server problems, or other interruptions that prevent the data from being transferred within the expected time frame.
102	The EC_Expire error indicates that a task or job has expired. This typically means that the task did not complete within a specified time limit and was therefore terminated. Expiration can occur due to various reasons, such as system constraints, resource limitations, or predefined timeout settings.
103	The EC_Aborted error means that a task or job was aborted. This typically means that the process was intentionally stopped before it could complete. This could happen for various reasons, such as a manual intervention by an operator, a system decision to halt the task due to certain conditions, or an error that necessitated stopping the process to prevent further issues.
105	The EC_Reassigned error indicates that a task or job has been reassigned due to re-brokerage. Re-brokerage typically means that the job was moved to a different resource or broker for execution, possibly due to issues or inefficiencies with the original assignment. This error helps in identifying and handling such scenarios in the system.
106	The EC_Retried error means that a task or job has been retried by the server. This typically happens when the initial attempt to complete the task was unsuccessful, and the server decides to retry the task in hopes of a successful completion on the subsequent attempt. This error code helps in tracking and managing retries within the system.
107	The EC_PilotRetried error indicates that a job was retried by the pilot. In distributed computing, a pilot is a lightweight job that manages the execution of tasks on a computing resource. When a task fails or encounters an issue, the pilot can decide to retry the task in an attempt to complete it successfully. This error code is used to signify that such a retry attempt has been made by the pilot.
111	The EC_EventServiceRetried error means that an event service job has been retried. This typically means that the job encountered an issue during its execution and was automatically retried by the system in an attempt to complete it successfully. This error code helps in identifying and handling scenarios where retries are necessary for event service jobs.
112	The EC_EventServiceMerge error indicates that there was an issue related to merging for the event service. This typically means that an event service job encountered a problem during the merge process, which is a step where individual event outputs are combined.
113	The EC_MergeFailed error means that a merge job has failed. This typically means that an operation intended to combine or merge data, files, or tasks did not complete successfully. The specific context of the failure would depend on the broader application logic and what the merge operation was intended to accomplish.
114	The EC_EventServiceMaxAttempt error indicates that the maximum number of attempts for an Event Service operation has been reached. This typically means that the system has tried to process an event multiple times but has failed to do so within the allowed number of retries. This error is used to signal that no further attempts will be made for this particular event, and it should be handled accordingly.
115	The EC_EventServiceWaitOthers error means that the system is doing

Examples

Log Analysis Example (“User”)

- Response from Gemini (free 2.5-flash model) for a job that failed with “remote file open timed out” for “non-expert users”
 - (Prompt contains request for both user and expert explanations)

1) Explanations and suggestions for non-expert users:

- * **What happened?** The job tried to access a file located on a remote server, but it took too long, and the connection timed out. Think of it like trying to load a website that never finishes loading.
- * **Why did it happen?** There are a few possibilities:
 - * **The remote server was busy or slow:** The server hosting the file might have been overloaded with other requests, causing delays.
 - * **Network problems:** There might have been issues with the network connection between the worker node and the remote server. This could involve temporary network outages, high network traffic, or firewall issues.
 - * **File system issues:** The file may have been stored on a filesystem with underlying issues that caused it to become inaccessible.
 - * **Configuration issues:** The error message "Info: /cvmfs mounted; do 'setupATLAS -d -c ...' to skip default mounts. Info: \$HOME mounted; do 'setupATLAS -d -c ...' to skip default mounts." indicates a potential problem with how the ATLAS software environment is set up.
- * **What can you do?**
 - * **Try re-running the job:** Often, these types of transient network or server issues resolve themselves. A simple retry might be enough.
 - * **Check the PanDA monitoring dashboard:** Look for any widespread problems or warnings related to the storage system or network.
 - * **Contact support:** If the problem persists or if you see other jobs failing with similar errors, contact the PanDA support team or the resource provider. Provide them with the job ID and the error message.
 - * **Check to see if there are any known issues with network latency to the LPC site:** Use standard tools (e.g. `ping`, `tracert`, `mtr`) to probe network latency to the LPC site.
 - * **Check to see if the /cvmfs filesystems are healthy:** Problems with the CVMFS client can cause hangs in remote file access.

Log Analysis Example (“Expert”)

- Gemini response for same job (remote file open timed out) for “experts” with more details and suggestions
- Plan: user could select “User” or “Expert” response in PanDA monitor

2) Explanations and suggestions for experts and/or system admins:

* **Root Cause Analysis:** The error message “PilotException: error code: 1368, message: Remote file open timed out” indicates a timeout during remote file access. The accompanying details “Info: /cvmfs mounted; do ‘setupATLAS -d -c ...’ to skip default mounts. Info: \$HOME mounted; do ‘setupATLAS -d -c ...’ to skip default mounts.” are crucial.

* **CVMFS and ‘\$HOME’ Mounts:** The message suggests that the default CVMFS and ‘\$HOME’ mounts might be interfering with the remote file access, especially within a containerized environment (Apptainer). The ‘setupATLAS -d -c ...’ command disables these default mounts. This implies that the job might be trying to access files that are shadowed by the default CVMFS or ‘\$HOME’ mounts, leading to resolution problems or conflicts that cause delays and timeouts.

* Possible Causes and Solutions:

1. **Incorrect ATLAS Set up Inside Container:** The container might not be correctly configured to access the necessary ATLAS software and libraries.

* **Solution:** Ensure the ‘setupATLAS’ script (or equivalent environment setup) is executed “within” the container with the ‘-d -c ...’ flags to avoid default mounts that can conflict with the intended file access. This might require modifying the payload execution script or the container image itself.

2. **Network Latency to IN2P3-LPC Storage:** The logs show the job attempting to access a file at ‘root://d1cgse01.in2p3.fr:1094//dpm/in2p3.fr/home/atlas/atlasdatadisk/rucio/data18_13TeV/a74b/DAOD_PHYS.43636385_000195.pod.root.1’. This points to a specific storage endpoint (IN2P3-LPC). High latency or intermittent network connectivity issues to this site will cause timeouts.

* Solution:

- * **Network Monitoring:** Monitor network performance between the worker node and ‘d1cgse01.in2p3.fr’. Use tools like ‘ping’, ‘traceroute’, ‘mtr’, and ‘iperf’ to identify latency, packet loss, or routing problems.
- * **Firewall Rules:** Verify that firewall rules are not blocking or throttling traffic between the worker node and the IN2P3-LPC storage endpoint. Pay attention to port 1094, which is used by the ROOT protocol.
- * **Storage System Health:** Contact the IN2P3-LPC storage administrators to check the health and performance of their storage system (dCache) and the ROOT service.
- * **Routing:** Check routing and ensure it is optimized for high performance.

3. **ROOT Protocol Issues:** The job uses the ROOT protocol (‘root://’). There might be problems with the ROOT client configuration or version compatibility on the worker node or within the container.

* Solution:

- * **ROOT Version:** Ensure that the ROOT version used in the job and the container is compatible with the storage server.
- * **ROOT Configuration:** Check the ROOT configuration files (e.g., ‘.root.c’) for any settings that might be causing connection problems or timeouts. Pay particular attention to settings related to network buffering and connection timeouts.

4. **CVMFS Interference with file access:** There may be issues with the worker node accessing CVMFS.

* Solution:

- * **Check CVMFS client:** Verify that the CVMFS client is properly configured and running on the worker node.
- * **CVMFS Version:** Ensure the CVMFS version is up-to-date and compatible with the ATLAS software stack.

5. **Rucio Service Availability:** Although less likely given the detailed error, verify the Rucio service (‘rucio-bprod.cern.ch’) is healthy. While the logs show successful Rucio interaction for trading, issues could still arise later.

* **Solution:** Check the Rucio status pages and dashboards for any reported incidents.

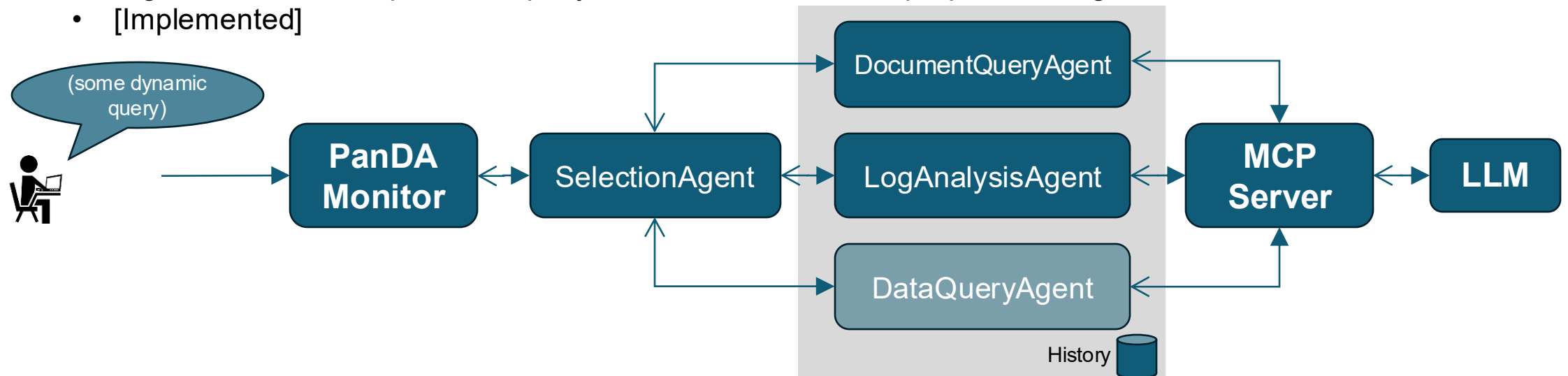
* Debugging Steps:

1. **Reproduce the Problem:** Try to reproduce the problem manually on the worker node or within a similar container environment. This will allow you to isolate the issue and test potential solutions.
2. **Increase Verbosity:** Increase the logging verbosity of the Pilot, ROOT, and any other relevant software components to gather more detailed information about the error.
3. **Packet Capture:** Use ‘tcpdump’ or ‘Wireshark’ to capture network traffic between the worker node and the storage server. This can help identify network latency, dropped packets, or other network-related problems.
4. **Consult Pilot and ATLAS Documentation:** Review the Pilot and ATLAS documentation for troubleshooting tips and known issues related to remote file access and containerization.

By systematically investigating these areas, you should be able to pinpoint the root cause of the “Remote file open timed out” error and implement the appropriate solution.

Selection Agent

- We might (probably) offer log file analysis of failed jobs via button on failed job pages, but ..
- We should not expect chatbot to be for only documentation queries
 - What if a user asks for dynamic information in chatbot query?
 - “Why did my job NNN fail?”, “How many of my jobs in task MMM failed?”, “Why did they fail?”
- Clumsy to implement natural language interpretation in chatbot when it could be done by LLM
 - But LLM doesn’t know about PanDA jobs and tasks!!
- Enter the “Selection Agent”
 - LLM can be used on multiple levels for several type of queries
 - E.g. use LLM to interpret user query and ask it to select the proper MCP agent
 - [Implemented]



Data Query Agent

- In early development: The “Data Query Agent” is a tool for querying about downloaded metadata, e.g.
 - Queue data (*information about queues*; which copy tool is used where, how much corepower does queue X have?)
 - Task information (*information about tasks*; how many jobs failed in task N?)

The “Data Query Agent” used dynamically via the “Selection Agent”

```
$ python3 -m agents.selection_agent --question "how many jobs failed in task 45607163" --model gemini --session-id 1
..
2025-07-21 23:25:45,204 INFO:Answer:
{ 45607163:
    { 'description':
        'The PanDA task is in progress. 21 jobs have finished, but 7 jobs have failed.',
      'problems':
        'Some jobs failed due to payload execution failures (error codes 1305, 5318),
        invalid configuration of a reduction job (error code 1368), wrapper fault (error code 1361) and
        remote file could not be opened(error code 2001).'
    }
}
```

MCP Server for Claude Desktop

- Ask PanDA partly ported as MCP server(s) for Claude Desktop
 - One server for log analysis
 - Async downloads metadata and pilot log from PanDA, constructs prompt and returns to LLM for interpretation
 - Another server for questions about site information
 - Server downloads metadata from CRIC, returns to LLM
 - (Actually standard Ask PanDA doesn't know how to do this yet)
- Can currently handle prompts like
 - *“Show me all the queues that are using rucio as copytool”* [metadata]
 - *“Display the corepower for all queues”* [metadata]
 - *“What copytool does BNL use?”* [metadata]
 - *“What is the status of job 6709690286?”* [metadata]
 - *“Why did job 6715819744 fail?”* [log analysis]
- Different types of prompts have to be supported individually
- ***Slow*** development using free Claude account
 - Either runs out of credits or dialogs become too long too quickly
 - But it is .. free

PN Why did job 6715819744 fail?

I'll analyze job 6715819744 to determine why it failed. Let me download the metadata and examine the failure details.

G panda_job_analyze_failure

Based on the failure analysis, job 6715819744 failed due to a **file transfer timeout during stage-out**. Here are the key details:

Failure Summary

- **Job Status:** Failed
- **Primary Cause:** File transfer timeout (Pilot Error Code 1152)
- **Site:** UKI-NORTHGRID-SHEF-HEP (Sheffield, UK)

Specific Error Details

The job failed during the stage-out phase when trying to transfer the output file `output.1.2abc361d-8752-479b-a4ea-303d72221559_61142.pool.root` from the computing site to the storage destination `UKI-NORTHGRID-MAN-HEP-CEPH_DATADISK`.

Error Message: "File transfer timed out during stage-out: copy command timed out: TimeoutException: Timeout reached, timeout=310 seconds"

Job Execution Details

- The job itself executed successfully (transformation exit code: 0)
- CPU consumption was only 19 seconds, indicating the actual processing completed quickly
- The job ran for about 58 minutes total, but most of this time was spent waiting or transferring files
- Memory usage was normal (max RSS: 67MB)

What Happened

1. The job successfully processed the input file (2.5GB DAOD_PHYS file)
2. The actual computation completed without errors
3. During the final stage-out phase, the file transfer to storage timed out after 310 seconds
4. The Rucio copy tool failed to complete the transfer within the timeout period

This is typically a network or storage system issue rather than a problem with the job's code or 15 configuration. The job could be retried, as it has 2 more attempts remaining (currently on attempt 1 of 3).

On the Shorter Term

- Further development of MCP agents
 - Incl. analysis of (esp.) user payload output
 - Agent will automatically download payload log for certain errors
 - Check existing MCP agents/servers on the market
 - E.g. <https://github.com/modelcontextprotocol/servers> (Grafana MCP server looks interesting)
 - Investigate how to interact with such servers via existing agents/tools
 - Persist log analyses, see if reuse is possible (compare explanations for same problems)
 - Complete query history in agents (other than “Document Query Agent”)
 - Compare log analyses using different LLMs
- Select front end UI
 - *OpenWebUI* suggested, others include *LibreChat* and *AnythingLLM*
- Current development version of Ask PanDA is tested against ATLAS tasks and jobs
 - More or less trivial to extend to other experiments (it’s a matter of source URLs)

On the Longer Term

- Scrape and filter discussions about user problems from mailing list(s)
 - Store in vector store
 - “Document Query Agent” can extract relevant discussion for known problems/resolutions
 - Note: currently only relevant for ATLAS (no other data is available), but good to get experience – eventually this will be an ongoing activity
- MCP server optimizations
 - Multi-user (queries) testing
 - Vector store updates in many-user environment vs many documents in resources folder
- Fine-tuning on selected open weight models (SLAC project)
 - Examples of open weight models include Llama, Mistral, **Gemma**
 - Using opportunistic GPU resources
 - Fine tuned models will have basic knowledge [better than commercial LLMs] about PanDA
 - Will also have knowledge about Ask PanDA to e.g. select which agent is to be used
- Continue development of Claude agents/servers
- Test MCP agents with new version of PanDA Monitor as soon as it is available