BATCH MONITORING

December 2024 v1

https://www.linkedin.com/in/eduardoagarciamendoza/

AGENDA

1. Introduction
2. Batch Processing
3. Monitoring Elements
4. Scheduling Algorithms
5. Batch Monitoring Tools
    - 5.1 Autosys
    - 5.2 Airflow
6. Examples

What is a batch job?

What are the characteristics of a Batch Job?

Batch Jobs

"A predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system"

Batch Jobs are:

- Non-interactive
- Scheduled
- Sequential
- Automated
- Predictable

But can also be:

- Long running
- Resource Intensive
- Dependent
- Error Handling intensive

(*IBM Documentation*, n.d.)

Batch job Scheduling Tools

```
# * * * * *  command to execute
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ └──── day of week (0 - 7)
# │ │ │ └────── month (1 - 12)
# │ │ └──────── day of month (1 - 31)
# │ └────────── hour (0 - 23)
# └──────────── min (0 - 59)
```

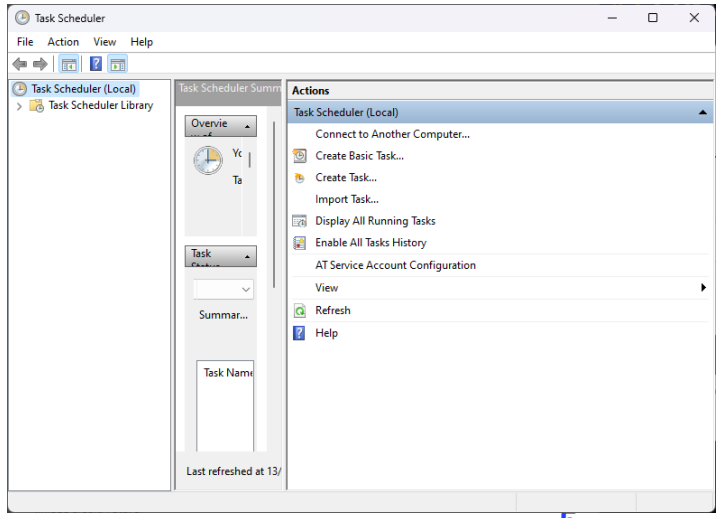
cron

Autosys

Jenkins

Control-M

IBM Workload Scheduler



When to use Batch Processing?

Use the correct approach

Batch Jobs

- Already the status quo
- Compatible with most legacy systems
- Program them to execute when compute power is available
- Excellent for complex analysis or intensive data processing

Stream processing

With the advent of Big Data, some companies focus on real time analysis. Imagine that some data will be out of date or stale before it can be acted on (ie, stock market or trading information). The emphasis on real time means:

- Data collected might be only relevant for a short time
- Analysis will probably be more shallow
- Ability to be agile and act upon immediately or in a short amount of time is paramount.

(Segner, 2023)

Generic Starting Conditions

Time and Date

- Monday to Friday at 1 am
- Every hour
- Every 15 minutes

Event Trigger

- Completion of a Job
- Data Imported
- User login

File Based

- Changes in filesystem. Creation, modification or deletion of a file/directory

Job Dependencies

- **Data Pipelines** usually have jobs set to start after one is completed

API calls

- When available and in complex environments you can set up a batch job initiated by an API call from another service

Resource Availability

- You can set up non-critical jobs to be scheduled once a system is available, ie DB usage has dropped for a certain amount of time or CPU usage falls below a threshold.

Generic Job Status Conditions

There are as many Job status conditions as Monitoring and Scheduling Tools exists. There is a list below of basic status conditions that mean more less the same across platforms.

- Not Started – Pending, Inactive, Runnable
- Running
- On Hold – held, on ice, on hold
- Success – SU, Succeded
- Failure – FA, Failed

- https://techdocs.broadcom.com/us/en/ca-enterprise-software/intelligent-automation/autosys-workload-automation/12-0/scheduling/ae-scheduling/manage-your-jobs/job-states.html

Monitoring Purpose

Batch jobs are usually used in data transformation, report generation, backups, hygiene, and synchronization.

- Ensure job starts at scheduled time and ends in SU status
- Monitoring job's progress
- Monitoring job's performance
- Addressing any problems that arise during execution.
- Verifying output of the job is correct and is complete.

Scheduling Algorithms

These depend on your monitoring tools.

Algorithm	Definition	Advantages	Disadvantages
FCFS (First Come, First Served)	Processes are executed in the order they arrive.	Simple to implement, fair in the sense that jobs are processed in the order they arrive.	Can lead to poor performance if long jobs arrive early and block shorter jobs (convoy effect).
SPF/SJF (Shortest Processing Time First / Shortest Job First)	The process with the shortest estimated processing time is executed first.	Minimizes average waiting time.	Requires knowing the processing time in advance, which is often difficult. Can lead to starvation for long jobs if short jobs keep arriving.
RR (Round Robin)	Each process is given a fixed time slice (quantum) to execute. If a process doesn't complete within its time slice, it's moved to the back of the ready queue.	Provides fair CPU allocation, responsive to interactive processes.	Performance depends heavily on the choice of time quantum.
EASY (Extensible Argus System)	Designed for parallel batch jobs. Starts with the first job and tries to fit other jobs into the gaps while waiting for resources.	Efficiently utilizes resources by overlapping I/O and CPU operations.	More complex to implement than simpler algorithms.
Priority Scheduling	Each job is assigned a priority, and higher-priority jobs are executed first.	Allows important jobs to be executed quickly.	Can lead to starvation for low-priority jobs.
LRF (Largest Ratio First)	Similar to priority scheduling, but the "priority" is calculated as a ratio of metrics (e.g., waiting time divided by processing time). The job with the largest ratio is run first.	Aims to balance fairness and efficiency.	The choice of metrics can significantly impact performance.

Pitfalls of Monitoring

	As a human, you will probably default to first address issues based on these two algorithms while monitoring. This is not good for your client.
FCFS	First Come, First Served
SPF	Smallest estimated Processing Time first

While FCFS and SPF seem efficient, they can create an 'alert fatigue' scenario. Engineers **will** focus on the earliest or easiest alerts, potentially missing deeper, systemic issues impacting batch performance. This is just human nature. You can also default to the oldest open issue in your monitoring queue.

This is a simplistic approach that will work most of the time. Relying solely on FCFS or SPF for alert prioritization risks delaying the resolution of critical batch issues, potentially impacting SLAs and business operations.

Using critical thinking as well as asking yourself "**what is the impact to the business if this job fails?**" will lead to better results than blindly addressing issues that may not be critical or better yet, you may actually prevent a production outage or make an early detection.

The Cost of Failure

How do batch failures impact our bottom line? Let's explore this by examining the ripple effects:

You have no idea? Ask this – What is the impact to production? Followed by, What is the impact to our Business Unit? Will we breach an SLA?

While handling a live issue:

- **Revenue Loss**: How do delays or errors translate to lost revenue opportunities?
- **Operational Disruption**: What key business processes are impacted, and how does this hinder our ability to function? (e.g., order fulfillment, reporting, customer service)
- **Regulatory Compliance**: Could these failures lead to non-compliance with industry regulations or legal requirements?

- On a post-mortem:

- **Reputational Damage**: How do batch issues affect customer trust and brand perception? (e.g., delayed payments, inaccurate information)
- **Opportunity Cost**: What could we achieve with the time and resources spent firefighting batch issues? (e.g., innovation, strategic initiatives)

Top Monitoring Tools

(Raj, 2022)

Autosys

Key Components:

•Event Server (manages events and dependencies)
•Event Processor (executes jobs based on events)
•Shadow Processor (backup for Event Processor)
•Client Utilities (command-line tools for managing jobs)
•GUI (web-based interface for monitoring and control)

Core Concepts:

•Jobs (units of work)
•Machines (where jobs run)
•Boxes (logical groupings of jobs)
•Schedules (define when jobs run)
•Dependencies (relationships between jobs)
•Attributes (control job behavior)

Example Job Definition:

```
insert_job: my_job machine:
my_machine command:
my_script.sh
date_conditions: 1
start_times: "00:00"
```

This defines a job named "my_job" that runs "my_script.sh" on "my_machine" every day at midnight.

Autosys commands

Category	Command	Description
Job Control	sendevent -E STARTJOB -J job_name	Starts a job
Job Control	sendevent -E FORCE_STARTJOB -J job_name	Force starts a job, overriding dependencies
Job Control	sendevent -E KILLJOB -J job_name	Kills a job
Job Control	sendevent -E JOB_ON_ICE -J job_name	Deactivates job; doesn't impact dependent jobs
Job Control	sendevent -E JOB_OFF_ICE -J job_name	Reactivates job; runs at next scheduled time
Job Control	sendevent -E JOB_ON_HOLD -J job_name	Prevents the job from running; holds dependent jobs
Job Control	sendevent -E JOB_OFF_HOLD -J job_name	Releases a job from hold; runs immediately if conditions are met
Job Control	sendevent -E CHG_JOB_PRIORITY -J job_name -p new_priority_value	Changes job priority; higher priority gets resource preference
Job Control	sendevent -E JOB_BEGIN -J job_name	Manually forces a job to start (use with caution)
Job Control	sendevent -E JOB_END -J job_name	Manually forces a job to end
Job & Machine Info	autorep -J job_name	Displays job details
Job & Machine Info	autorep -J job_name -q	Displays the JIL for the job
Job & Machine Info	autorep -M machine_name	Displays machine details
Job & Machine Info	autorep -G global_variable_name	Displays the value of a global variable
Job & Machine Info	autostatus -J job_name	Displays the status of a job
System Status & Logs	autosyslog -J job_name	Displays log messages for a specific job

Airflow

Key Components:

•Directed Acyclic Graphs (DAGs): Workflows are defined as DAGs, where nodes represent tasks and edges represent dependencies between them.
•Operators: Building blocks of DAGs. They perform individual tasks (e.g., BashOperator executes bash commands, PythonOperator executes Python code).
•Scheduler: Automatically runs your tasks on an array of workers at the right time, while following the specified dependencies.
•Web UI: Provides a rich user interface to visualize pipelines, monitor progress, and troubleshoot issues.

Benefits:

•**Dynamic & Scalable:** Easily create complex workflows with dynamic pipelines that scale to handle large volumes of data.
•**Extensible:** Large library of community-built operators and integrations with various tools and services.
•**Observability:** Monitor workflows in real-time, track progress, and receive alerts on failures.

Airflow

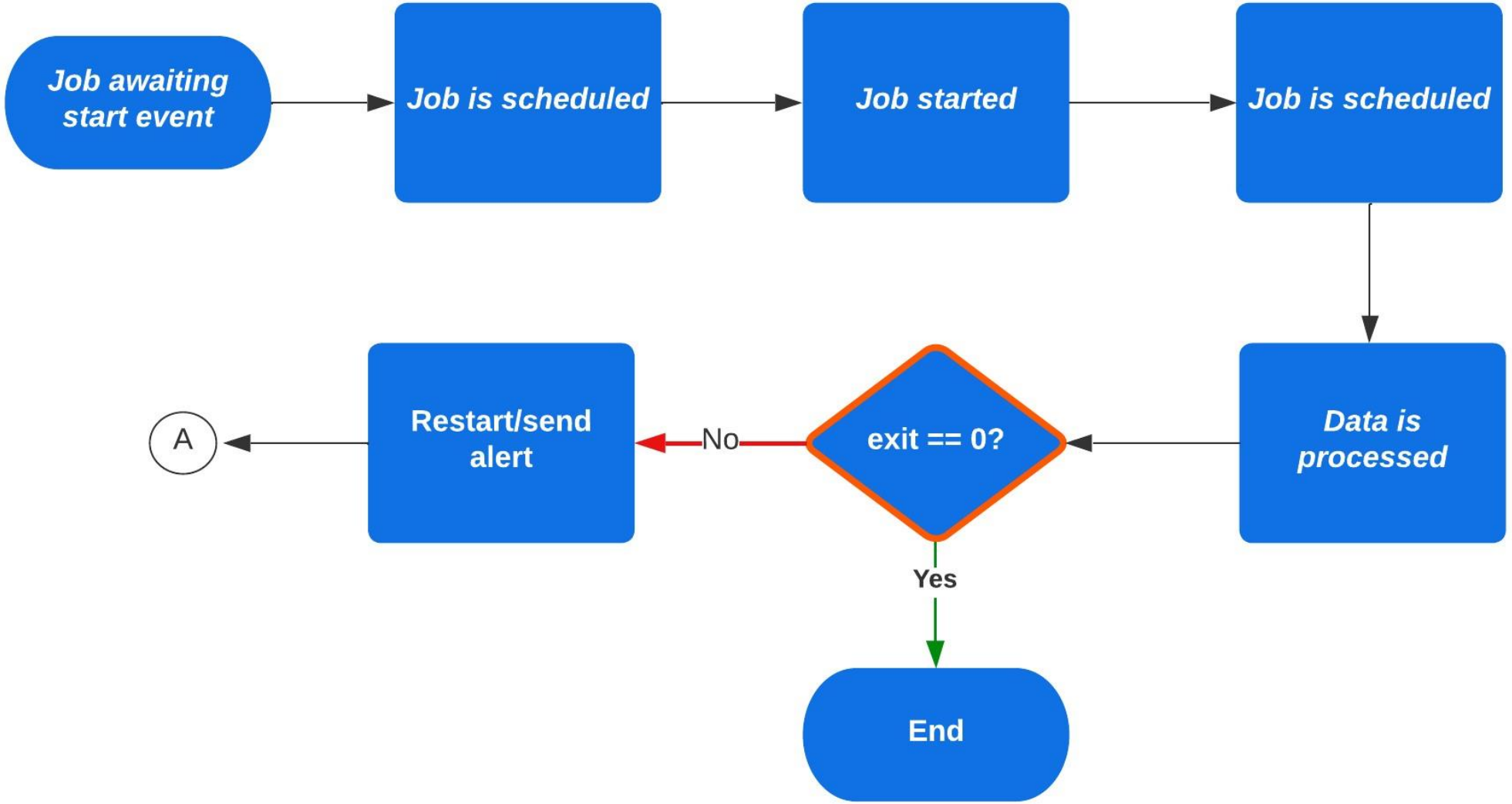
```python
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime

with DAG(
    dag_id='my_dag',
    start_date=datetime(2023, 1, 1),
    schedule_interval='@daily',
    catchup=False
) as dag:

    t1 = BashOperator(
        task_id='print_date',
        bash_command='date',
    )

    t2 = BashOperator(
        task_id='sleep',
        bash_command='sleep 5',
    )

    t1 >> t2
```

This defines a DAG that runs daily, starting from January 1, 2023. It has two tasks: print_date prints the current date, and sleep pauses for 5 seconds. t1 >> t2 defines the dependency, ensuring print_date runs before sleep.

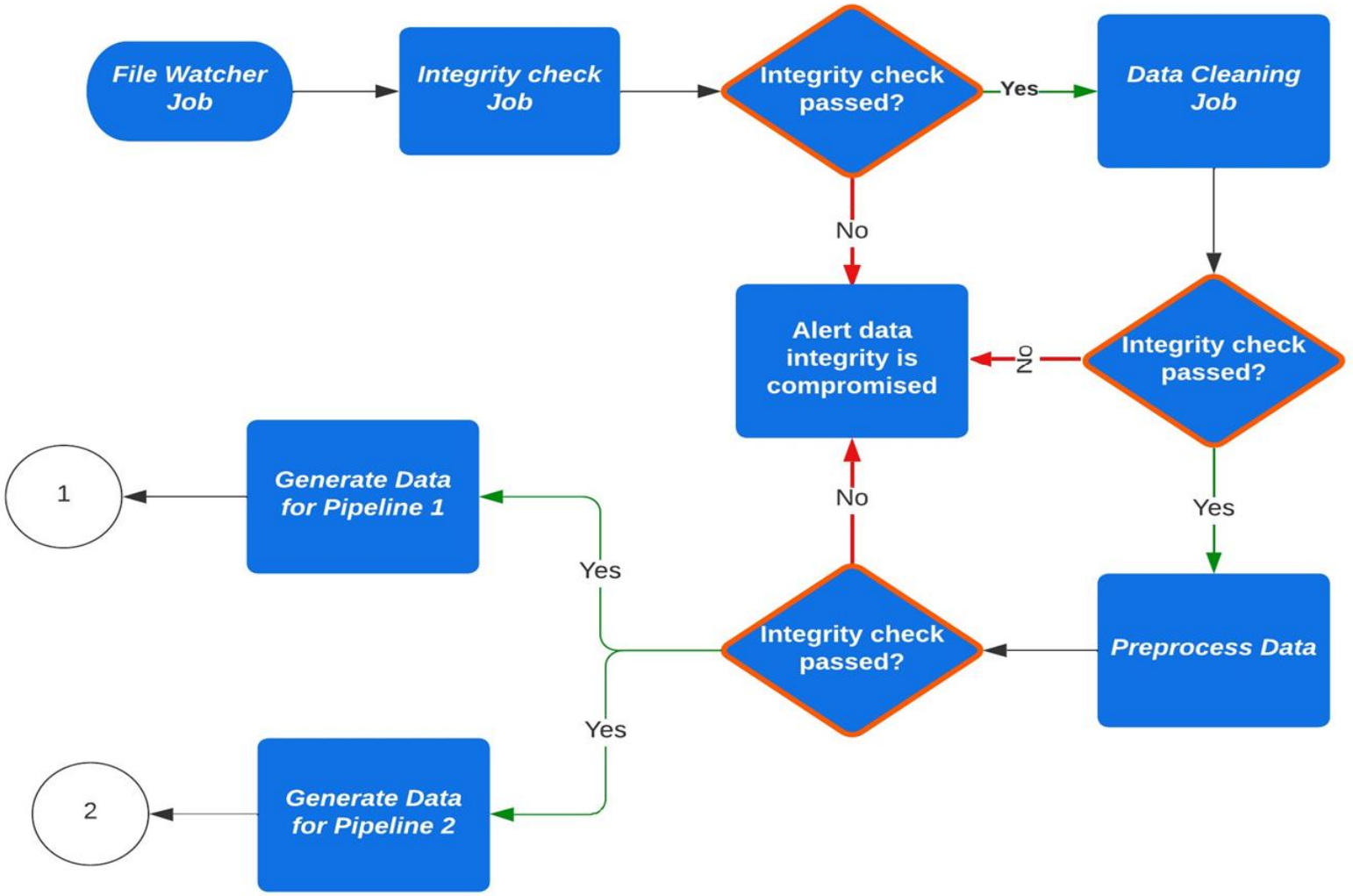
Airflow commands

Category	Command	Description
Initialization & Database	airflow db init	Initializes the Airflow metadata database
	airflow db upgrade	Upgrades the database to the latest schema
	airflow db reset	Resets the database (use with caution!)
Users & Roles	airflow users create	Creates a new user
	airflow roles create	Creates a new role
	airflow users list	Lists all users
Connections	airflow connections add	Adds a new connection
	airflow connections list	Lists all connections
	airflow connections delete	Deletes a connection
DAGs & Tasks	airflow dags list	Lists all DAGs
	airflow dags show <dag_id>	Displays the structure of a DAG
	airflow tasks list <dag_id>	Lists all tasks in a DAG
	airflow tasks test <dag_id> <task_id> <execution_date>	Tests a single task
Scheduling & Execution	airflow dags trigger <dag_id>	Triggers a DAG run
	airflow dags backfill <dag_id>	Runs a DAG for a specified date range
	airflow scheduler	Starts the scheduler to monitor and trigger tasks
	airflow webserver	Starts the webserver for the Airflow UI
Troubleshooting & Monitoring	airflow logs <dag_id> <task_id> <execution_date>	Shows logs for a task instance
	airflow task state <dag_id> <task_id> <execution_date>	Gets the state of a task instance
	airflow clear <dag_id>	Clears task instances for a DAG
Plugins & Providers	airflow plugins list	Lists installed plugins
	airflow providers list	Lists installed providers

Generic Batch Job Workflow



Generic Data Pipeline Workflow



References

- Raj, J. (2022, December 23). *Top 10 Monitoring and observability tools in 2022 for SRE (Site reliability engineering) - DevOpsSchool.com*. DevOpsSchool.com. https://www.devopsschool.com/blog/top-10-monitoring-and-observability-tools-in-2022-for-sre-site-reliability-engineering/
- IBM Documentation. (n.d.). https://www.ibm.com/docs/en/i/7.2?topic=types-batch-jobs
- Segner, M. (2023). Intro To Batch Vs Stream Processing - With Examples. *Monte Carlo Data*. https://www.montecarlodata.com/blog-stream-vs-batch-processing/

THANK YOU

Eduardo Alejandro Garcia Mendoza