Nextflow Data Transfer Pipeline

Overview

This Nextflow pipeline automates the preparation and transfer of large genomic datasets, including file compression, checksum generation, batch transfer list creation, and error handling. It is designed for efficient and secure data movement between local workstations and high-performance computing (HPC) environments using tools like Globus.

Key Features

- Automated directory structure setup for raw and processed data
- Batch file compression to reduce transfer times
- Checksum generation for data integrity verification
- Batch transfer list preparation for efficient Globus transfers
- Real-time logging and error handling
- Email notifications on pipeline completion or failure

Project Directory Structure

```
Genomic_Data_Transfer_Pipeline/
├── Raw_Data/             # Uncompressed raw data files
├── Processed_Data/       # Processed data ready for analysis
├── Archives/             # Compressed data for transfer
├── Logs/                 # Pipeline logs
├── Checksums/            # Checksum files for data validation
├── Scripts/              # Nextflow scripts
└── Config/               # Pipeline configuration files
```

Setup Instructions

1. **Clone the repository**

```
git clone https://github.com/yourusername/Genomic_Data_Transfer_Pipeline.git
cd Genomic_Data_Transfer_Pipeline
```

2. **Create the Project Directory**

```
bash create_project_structure.sh
```

3. **Configure Nextflow**

- Update the `nextflow.config` file in the `Config/` directory if needed.

4. **Run the Pipeline**

```
nextflow run Scripts/main.nf –c Config/nextflow.config
```

Usage Notes

- **Input Files**: Place raw `.fastq` files in the `Raw_Data/` directory.
- **Output Files**:
  - Compressed files will be stored in `Archives/`
  - Checksum files will be stored in `Checksums/`
  - Batch transfer list will be created as `transfer_list.txt`
  - Logs will be written to the `Logs/` directory

Monitoring and Notifications

- Pipeline status is logged to `pipeline.log` in the `Logs/` directory.
- Email notifications are sent on pipeline completion or failure.

Error Handling

- The pipeline captures and logs all exceptions, terminating on critical errors.
- Check `pipeline.log` for detailed error messages.

Real-World Example

Sample Test Datasets

For realistic testing, you can generate small simulated datasets using the following commands:

```
mkdir -p Raw_Data
for i in {1..3}; do
  echo -e "@SEQ_ID_$i
GATTTGGGGTTTCCCAGTTGAGTCT
+
IIIIIIIIIIIIIIIIIIIIIIIII" > Raw_Data/Sample$i.fastq
done
```

This will create three small **.fastq** files in the **Raw_Data/** directory, each containing a simulated DNA sequence.

Larger Test Data (WGS, RNA-seq, Methylation)

For more realistic tests, consider downloading public datasets:

- **WGS Data**: Use the 1000 Genomes Project data (~100 GB per sample)
- **RNA-seq Data**: Use public datasets from GTEx or TCGA (~10 GB per sample)
- **Methylation Data**: Use data from the ENCODE or TCGA projects (~1 GB per sample)

Once downloaded, place these files in the **Raw_Data/** directory before running the pipeline.

To test the pipeline with sample data, you can use the following directory structure and simulated files:

Test Directory Structure

```
Genomic_Data_Transfer_Pipeline/
├── Raw_Data/
│   ├── Sample1.fastq
│   ├── Sample2.fastq
│   └── Sample3.fastq
├── Processed_Data/
├── Archives/
├── Logs/
├── Checksums/
├── Scripts/
└── Config/
```

Generating Test Data

You can quickly create test data with the following commands:

```
mkdir -p Raw_Data
for i in {1..3}; do
  echo "@SEQ_ID
GATTTGGGGTTTCCCAGTTGAGTCT" > Raw_Data/Sample$i.fastq
done
```

Running the Pipeline

Once the test data is in place, run the pipeline with:

```
nextflow run Scripts/main.nf -c Config/nextflow.config
```

Expected Outputs

- **Compressed Files**: `Archives/Sample1.tar.gz`, `Sample2.tar.gz`, `Sample3.tar.gz`
- **Checksum Files**: `Checksums/Sample1.tar.gz.sha256`, `Sample2.tar.gz.sha256`, `Sample3.tar.gz.sha256`
- **Transfer List**: `transfer_list.txt`
- **Logs**: `Logs/pipeline.log`

Advanced Usage Examples

For larger projects or more complex data transfer scenarios, consider these advanced usage patterns:

Parallel Transfers for Large Datasets

For projects involving hundreds of samples, you can parallelize the compression and checksum steps to reduce runtime:

```
nextflow run Scripts/main.nf -c Config/nextflow.config -with-dag flowchart.png -resume -with-report report.html
```

This command will:

- Generate a workflow DAG (`flowchart.png`) to visualize the pipeline structure.
- Create a detailed execution report (`report.html`).
- Generate a timeline of task execution (`timeline.html`).

Handling Multi-Project Transfers

For projects with multiple cohorts or centers, organize your data into subdirectories:

```
Genomic_Data_Transfer_Pipeline/
├── Cohort1/
│   └── Raw_Data/
│       ├── Sample1.fastq
│       └── Sample2.fastq
├── Cohort2/
│   └── Raw_Data/
│       ├── SampleA.fastq
│       └── SampleB.fastq
├── Archives/
├── Logs/
├── Checksums/
├── Scripts/
└── Config/
```

Modify the Nextflow script to process each cohort separately or in parallel for better scalability.

Automated Integrity Checks

Integrate checksum verification as part of the pipeline to ensure data integrity:

```
for file in Checksums/*.sha256; do
  sha256sum -c $file || echo "Checksum failed for $file"
done
```

This script will validate each compressed file against its checksum, ensuring data integrity after transfer.

Real-World Case Studies

Case Study 1: Large-Scale Cancer Genomics Project

Scenario: A cancer research institute is transferring multi-terabyte datasets, including WGS, RNA-seq, and methylation data, from local sequencing centers to a central HPC cluster for joint analysis.

- **Data Types**:
  - WGS (FASTQ) – ~3 TB per patient
  - RNA-seq (FASTQ) – ~200 GB per patient
  - Methylation (IDAT) – ~100 GB per patient
- **Challenges**:
  - High data volume (~300 TB for 100 patients)
  - Data integrity and error detection
  - Scalability for multi-center collaboration
- **Recommended Approach**:
  - Use compressed archives to reduce transfer times.
  - Automate data integrity checks with checksum verification.

- Use Globus for fast, secure data transfers.

Case Study 2: Multi-Center GWAS Study

Scenario: An international consortium is conducting a GWAS to identify genetic risk factors for a rare disease, requiring frequent data exchanges between collaborators.

- **Data Types**:
  - Genotyping arrays (PLINK files) – ~50 GB per cohort
  - WES (FASTQ) – ~500 GB per cohort
  - Clinical phenotypes (CSV) – ~5 GB per cohort
- **Challenges**:
  - Data privacy and compliance (e.g., GDPR)
  - Network latency for international transfers
  - Efficient metadata management
- **Recommended Approach**:
  - Use region-specific endpoints to reduce latency.
  - Implement strict access controls and data encryption.
  - Use automated transfer scripts to reduce manual errors.

Case Study 3: Clinical Genomics Pipeline

Scenario: A clinical laboratory is transferring sequencing data to an HPC for rapid variant calling and clinical interpretation.

- **Data Types**:
  - WES (FASTQ) – ~50 GB per sample
  - Clinical reports (PDF) – ~1 MB per patient
- **Challenges**:
  - Regulatory compliance (e.g., HIPAA, GDPR)
  - Fast turnaround times for patient results
  - Detailed logging for regulatory audits
- **Recommended Approach**:
  - Use automated file validation and logging to reduce compliance risks.
  - Compress raw data to reduce transfer costs.
  - Use email and Slack notifications for real-time monitoring.

Globus Integration

Efficient data transfer is critical for large-scale genomic projects. Globus provides a secure, high-speed, and scalable platform for moving data between endpoints, including local workstations, HPC clusters, and cloud storage.

Setting Up Globus Endpoints

1. **Personal Endpoint Setup**
   - Install **Globus Connect Personal** on your local machine.
   - Create a Globus account if you don't have one.
   - Set up a personal endpoint for local data transfers.
2. **HPC Endpoint Access**
   - Identify the appropriate endpoint for your HPC system (e.g., **"NIH HPC Data Transfer"** for Biowulf).
   - Request access if needed.
3. **Testing the Connection**

- Use the Globus web interface to transfer a small test file between your personal and HPC endpoints.

Automated Batch Transfers

Once the pipeline generates the **transfer_list.txt** file, you can automate the data transfer with the following commands:

```
# Replace with your endpoint IDs
SRC_ENDPOINT="your_personal_endpoint_id"
DST_ENDPOINT="your_hpc_endpoint_id"

# Run the batch transfer
globus transfer "$SRC_ENDPOINT:/path/to/Genomic_Data_Transfer_Pipeline/Archives/" "$DST_ENDPOINT:/path/to/HPC_D
```

Security Considerations

- **Use Encrypted Endpoints** – Ensure data is encrypted during transfer.
- **Enable Access Controls** – Use Globus' sharing features to restrict access to sensitive files.
- **Monitor Transfer Status** – Set up automated alerts for failed or delayed transfers.

Post-Transfer Validation

Once the transfer is complete, validate the data using the generated checksum files:

```
for file in Checksums/*.sha256; do
  sha256sum -c $file || echo "Checksum failed for $file"
done
```