HPC Scaling Instructions

Overview

This document provides **concise instructions** for running the Gene Regulatory Network reconstruction pipeline on a high-performance computing (HPC) cluster using SLURM. The HPC workflow is optimized for scalable inference on large expression datasets and comprises three main scripts:

1. **`setup.sh`**: Prepares the environment (module loading, path exports) and installs required software (e.g., ARACNe, dependencies).
2. **`aracne_inference.sh`**: Submits ARACNe jobs in parallel, distributing TF-wise MI calculations across compute nodes.
3. **`post_processing.sh`**: Aggregates per-TF outputs, performs network consolidation, and executes enrichment testing.

Environment Setup: setup.sh

1. **Load required modules** (example for an Lmod system):

   ```
   module load R/4.0.5
   module load java/1.8.0
   ```

2. **Activate a conda environment** (optional) containing ARACNe-AP and Bioconductor packages:

   ```
   source activate grn_env
   ```

3. **Define environment variables**:

   ```
   export DATA_DIR="/path/to/your/repo/data"
   export OUTPUT_DIR="/path/to/your/repo/results"
   export ARACNE_JAR="/path/to/ARACNe.jar"
   ```

Run:

```
bash hpc/setup.sh
```

ARACNe Inference: aracne_inference.sh

Generates a SLURM job array to process each TF separately.

1. **Key variables** (edit at top of script):
   - `TF_LIST`: Path to `data/TF_list.txt`.
   - `EXPR_MATRIX`: Path to `data/expression_matrix_01.txt`.
   - `CNA_MATRIX`: Path to CNA file (or `NULL` if unused).
   - `OUTPUT_DIR`: Base path for storing per-TF outputs.
   - `ARACNE_JAR`: Path to ARACNe-AP JAR file.
2. **Submit job array**:

   ```
   sbatch hpc/aracne_inference.sh
   ```

   - This creates an array job where each task reads one TF from `TF_LIST` and runs ARACNe on that TF's expression profile.
   - **SLURM directives** (e.g., `#SBATCH --array=1-<numTFs>`, `--cpus-per-task=4`, `--mem=16G`) can be adjusted based on cluster specs.
3. **Per-TF Output**:
   - Each array task writes MI results to: `results/aracne/<TF>_MI.txt`.

Post-Processing: post_processing.sh

Consolidates MI files into a global network and performs downstream enrichment.

1. **Key variables** (edit at top):
   - `ARACNE_DIR`: Directory containing per-TF MI outputs (`results/aracne/`).
   - `ENRICH_GMT`: Path to `data/gene_sets.gmt`.
   - `GWAS_LOCI`: Path to `data/gwas_loci.txt`.
   - `OUTPUT_DIR`: Base path for final results.
2. **Merge TF–target pairs**:

   ```
   bash hpc/post_processing.sh --merge
   ```

   - Merges all `<TF>_MI.txt` into `results/ARACNe_edges.txt`.

3. **Run enrichment testing**:

   ```
   bash hpc/post_processing.sh --enrich
   ```

   - Uses R scripts or `clusterProfiler` within the shell script to generate `results/VSE_results.tsv`.

4. **Master Regulator Ranking**:

   ```
   bash hpc/post_processing.sh --rank
   ```

   - Aggregates network statistics and outputs `results/MR_ranking.tsv`.

Job Submission Example

A typical workflow may look like:

```
# 1. Setup environment
bash hpc/setup.sh

# 2. Run ARACNe inference (TF-wise job array)
sbatch hpc/aracne_inference.sh

# 3. After job completion, merge and post-process
bash hpc/post_processing.sh --merge
bash hpc/post_processing.sh --enrich
bash hpc/post_processing.sh --rank
```

Notes & Best Practices

- **Resource allocation**: Adjust `--mem` and `--cpus-per-task` according to expression matrix size and TF count.
- **Job dependencies**: Use `--dependency=afterok:<ArrayJobID>` when submitting post-processing steps to ensure ARACNe jobs complete successfully.
- **Logging**: Each script writes logs to `results/logs/`; monitor for errors.
- **Checkpointing**: For very large TF lists, split into smaller arrays or resume from specific TF indices.

Feel free to modify resource requests and paths to match your HPC environment. Ensure you cite ARACNe (Basso *et al.*, 2005) when using this pipeline in publications.