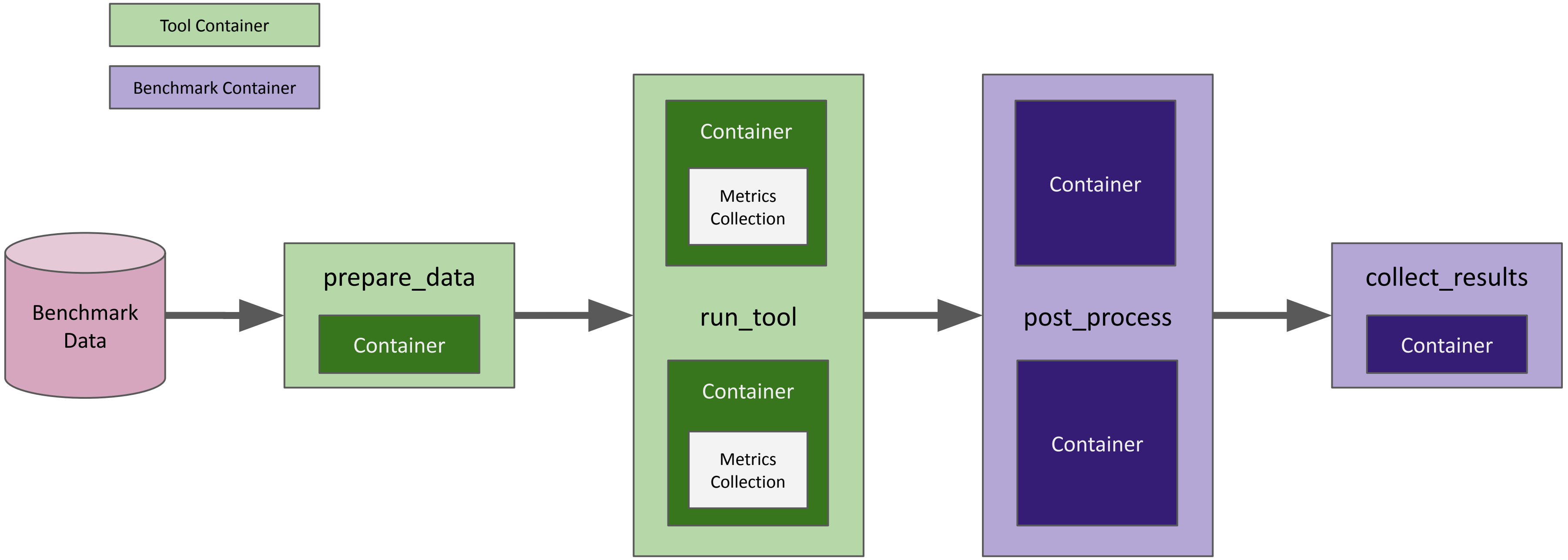
Transmark Harness Architecture

PSSS Codeathon, Sep. 2021

Terminology

- **Benchmark Developer** - a person who implements a particular benchmark dataset
- **Benchmark Tester** - the person or persons who create the scripts and container image necessary to benchmark a particular search tool
- **Search Tool** - a piece of software like HMMER or BLAST that searches a database for matches
- **Tool Container** - the user-provided image that knows how to prepare data and run the tool under benchmark

Workflow Diagram



Tool Container

Step 1 - prepare_data

- Transform data from generic input format to tool-specific format
- Train model, build index, etc.
- Implemented by benchmark tester

- Inputs: search targets and search space in generic input formats
- Outputs: tool-specific input files

Step 2 - run_tool

- Fan out to run the search tool being benchmarked
- Implemented by benchmark tester

- Inputs: tool-specific input files
- Outputs: search results in generic output format

Generic Input Formats

Transmark provides three inputs:

- DNA queries (MSAs) → `queries_dna.sto`
- Protein queries (MSAs) → `queries_protein.sto`
- Target sequences → `targets.fa`

Generic Output Format

Output is a tab-delimited table:

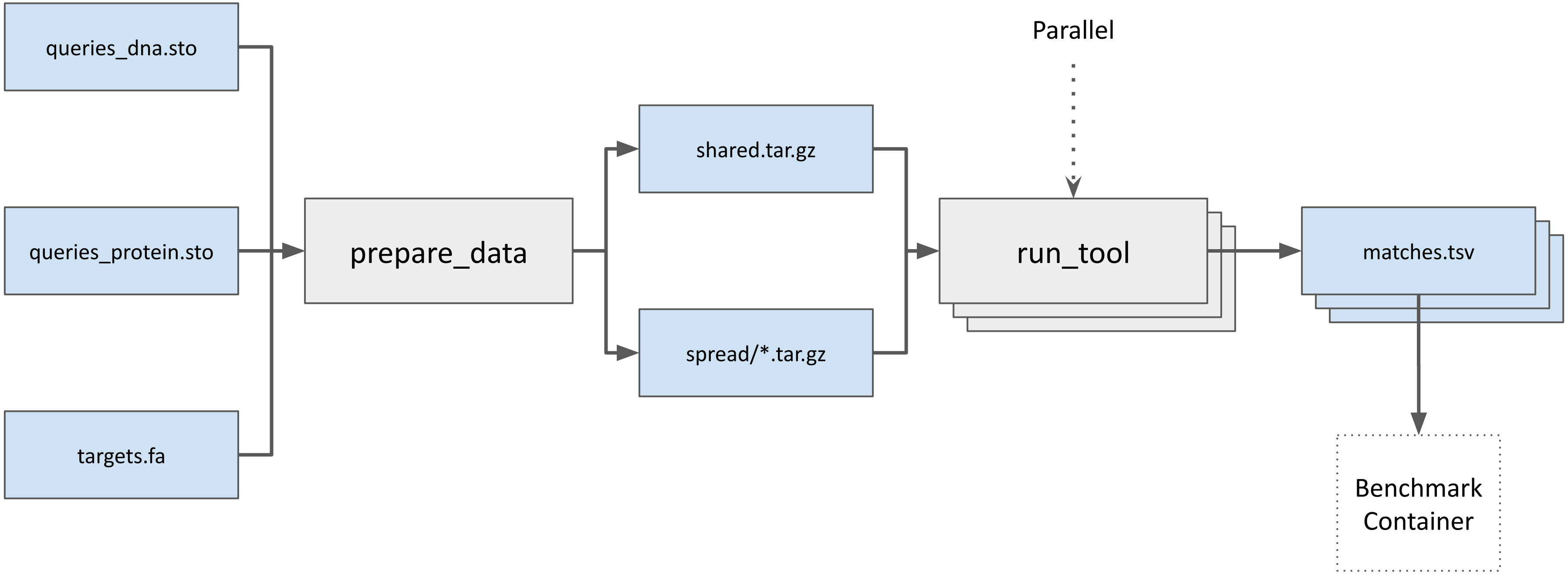
- E-value
- Score
- Hit start position (on chromosome)
- Hit stop position (on chromosome)
- Target name
- Query name

Tool Container API

```
prepare_data <dna> <protein> <targets> → (shared, spread…)

run_tool <shared> <spread> → search results…
```

Tool Container API Diagram



Benchmark Container

Step 1 - post_process

- Augment results in parallel
- Provided by the benchmark developer

- Inputs: search results in generic output format
- Outputs: augmented search results

Step 2 - collect_results

- Concatenate generic outputs
- Provided by the benchmark developer

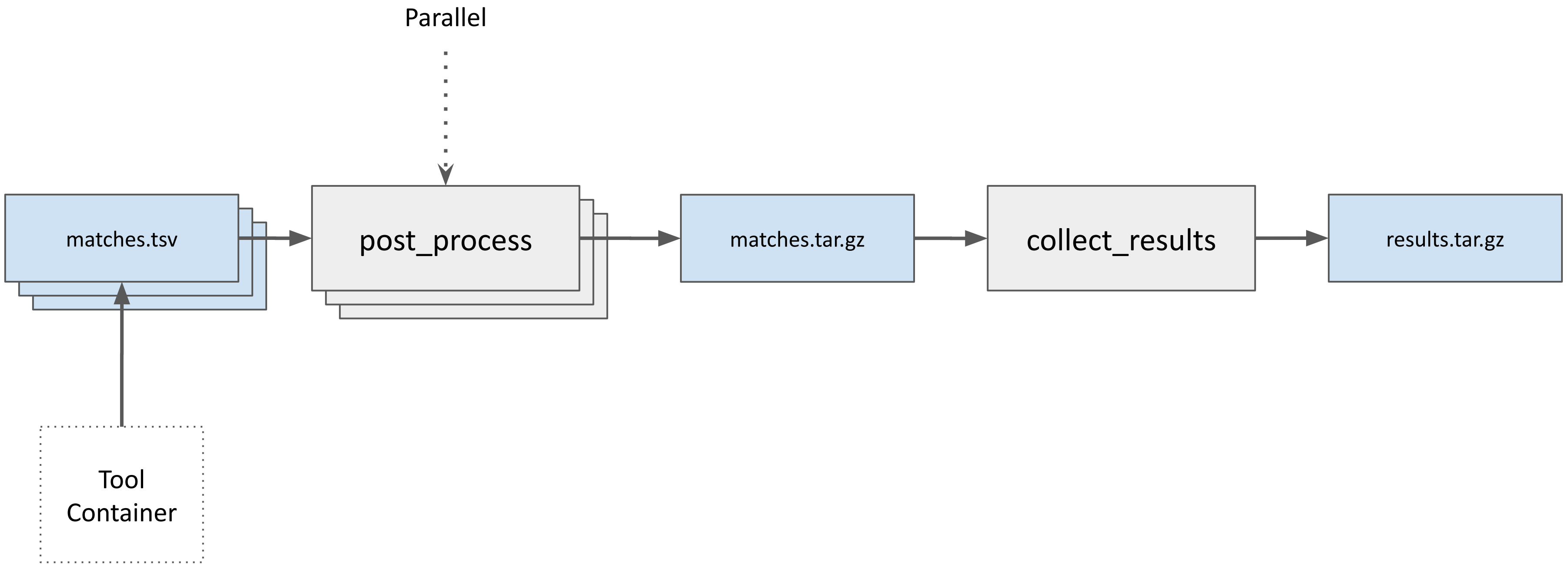
- Inputs: augmented search results
- Outputs: final results file

Benchmark Container API

```
post_process <search results> → augmented results…

collect_results <augmented results…> → final results
```

Benchmark Container API Diagram



Runner Application

Primary Purpose

Provide a simple command line application that abstracts over the implementation details of the tool containers, benchmarks, compute environment, and workflow tooling.

Why? To provide a consistent interface for running benchmarks.

Why? To promote use across the community.

Design

The runner will define formats for specifying tools and benchmarks and running those tools against the benchmarks.

Tools consist of a container image and a metadata file.

Benchmarks consist of a metadata file and, optionally, data files.

Data Management

Benchmark metadata may point to local or remote files.

Remote files are fetched either to the client or to the individual worker node, depending on how the backend works.

Nextflow can understand many URLs and “do the right thing”.

Tools

Metadata format:

```
{
      “name”: “mock-search”,
      “version”: “1.0.0”,
      “families”: [
            “transmark”
      ],
      “image”: “traviswheelerlab/bagel-mock-search:1.0.0”,
      “results”: {
            “matches”: “matches.tsv”
      }
}
```

Benchmark Families

Metadata format:

```
{
      “name”: “transmark”,
      “version”: “1.0.0”,
      “benchmark_data”: [
            “queries_dna”,
            “queries_protein”,
            “targets”
      ],
      “tool_results”: [
            “matches”
      ]
}
```

Benchmarks

Metadata format:

```
{
      “family”: “transmark”,
      “name”: “60% ID”,
      “version”: “1.0.0”,
      “image”: “traviswheelerlab/bagel-transmark:1.0.0”,
      “data”: {
            “queries_dna”: “https://osf.io/.../queries_dna.sto”,
            “queries_protein”: “https://osf.io/.../queries_protein.sto”,
            “targets”: “https://osf.io/.../targets.fa”
      }
}
```

Other Runner Goals

- Verify the correctness of tool metadata
- Assist in the development of new benchmarks
- Facilitate comparison of benchmark results
- Provide a vector for the distribution of benchmarks

Runner Backend

Backend Purpose

To organize computational resources in order to run benchmarks in an efficient, performant, and flexible manner.

Nextflow

The initial backend will be based on Nextflow.

This may be the only backend that ever exists, or it may not be.

It will likely be possible to define backends using Snakemake, WDL, Slurm, or a variety of other job and workflow management tools.

Future Work and Questions

Data Storage

Will NIH fund data storage going forward? We could use a single, default, S3 bucket for the “official” benchmarks if so.

Should we bundle / compress the data files? Doing so means they take up less space and are easier to transfer, but we can no longer download just one of them.

File Names

Should we use the names that come out of Transmark, or rename for greater clarity and generality (as described earlier)?

Results Collection

Should benchmarks be required to implement a collect_results step?

If so, should it be a Docker container?

If not, how should we do it?