To scale this pipeline to thousands of samples, I would primarily rely on sample-level parallelization, as each gVCF is processed independently. The current design already supports this through multiprocessing, and in a high-performance computing (HPC) environment, this could be extended by distributing samples across compute nodes using a workflow manager such as Snakemake or Nextflow or directly via SLURM job arrays.

For optimization, I would focus on minimizing I/O overhead by ensuring all gVCF files are bgzipped and indexed, enabling efficient streaming access without loading entire files into memory. Intermediate and final results would be stored in columnar formats such as Parquet to reduce disk usage and speed up downstream analyses. Profiling tools (e.g., cProfile or line_profiler) would be used to identify bottlenecks in variant parsing and QC filtering.

Benchmarking would involve measuring runtime, memory usage, and throughput as a function of sample count, using representative subsets of data. These benchmarks would guide the choice of parallelization strategy and resource allocation.

Finally, the pipeline would be containerized (e.g., Docker/Singularity) to ensure reproducibility and consistent performance across environments, allowing seamless execution on local machines, HPC clusters, or cloud platforms.