ForgeGIS™

Technical Brief

VERSION 1.0 · MAY 2026

Forged For Speed. Refined by Craft.

SEAGLASS FOUNDRY LLC

Executive Summary

ForgeGIS™ is a GPU-accelerated geospatial library with a native Model Context Protocol (MCP) server, distributed as pure-Java Maven artifacts with no first-party native code, no GDAL native bindings, and no separate native install step. It provides **220 user-facing operations** spanning terrain, bathymetry and ocean, hydrology, RF and visibility analysis, spectral and machine learning, spatial geometry, filtering and morphology, raster management and I/O, routing, point cloud, and temporal analysis — all callable directly from Java for programmatic pipelines, and exposed to AI agents through a dual-surface MCP server: a **pipeline DSL composing 133 of the operations** into multi-step workflows, and **54 curated intent tools** providing typed, single-call endpoints for common analytic tasks. As of the May 2026 validation run, we are not aware of another commercially-supported GPU-accelerated geospatial MCP server.

The library was designed around two architectural constraints that distinguish it from extension-based approaches to GPU GIS. First, the data plane is GPU-resident across operations rather than round-tripping through host memory or disk between stages — the design choice that makes multi-step agent-driven analysis viable at machine speed. Second, the deployment artifact is a single shaded Java jar with no first-party native code, no GDAL native bindings, and no separate native install step. GPU dispatch uses the JOCL OpenCL bindings, whose JNI bridge ships bundled inside the JOCL jar and extracts transparently at runtime; the system OpenCL runtime required for GPU access is standard system software bundled with GPU drivers. Compared to GDAL-based deployments, this eliminates the separate multi-vendor native library installation, GLIBC version-pinning, and per-platform shared library distribution that has historically dominated deployment cycles in regulated environments.

Performance, May 2026 validation run. ForgeGIS exceeds GDAL CLI by a geometric mean of **6.88× and a median of 6.51× across 143 catalog comparison records** — op × raster-scale measurements drawn from the 66 operations with direct GDAL CLI equivalents (127 wins, 16 losses; catalog-only slice excluding format-conversion round-trips). Multi-stage GPU-resident pipelines show speedups of **14.46× (TerrainPipeline, 4096^2) to 39.75× (WarpPipeline, 2048^2)** against equivalent sequential GDAL CLI chains. Across five real-world Copernicus and ETOPO datasets ranging from 2048^2 to 4096×4096 cells, per-op median speedups range from **13.76× to 36.94×**, with single-op maxes from 26.96× to 41.35×.

Validation and coverage. 78% of the validation surface — 731 of 935 records — tests operations with no GDAL CLI equivalent, primarily in terrain analytics, interpolation, ocean analysis, ML classification, viewshed, and trajectory work. These operations are validated against the appropriate domain references: JTS for Cartesian predicates and topology, PostGIS for spheroidal computations on the WGS84 ellipsoid, and numerical fixtures (analytic surfaces, bump targets, ramp gradients) where no external oracle exists. Outputs are verified deterministic across repeated runs in the validation harness, so the numerical guarantees scripted workflows depend on remain intact when the same operations are invoked by an AI agent.

ForgeGIS represents a generational shift in how geospatial analysis is delivered — from analyst-centric workflows to AI-native pipelines that operate at machine speed without sacrificing the determinism that regulated work requires. Seaglass Foundry is seeking strategic partners and distributors positioned to bring this technology to defense, intelligence, climate, and enterprise markets at scale.

What ForgeGIS Is

ForgeGIS is a geospatial compute substrate built for two audiences whose needs are converging: organizations that operate professional GIS pipelines today, and organizations that intend to expose geospatial analysis to AI agents tomorrow. It does not aspire to replace the analyst-facing GIS desktop. It replaces the layer underneath — the compute engine that runs the operations the analyst, the script, or the agent invokes.

The defining architectural commitment is that the data plane stays GPU-resident across operations. A multi-step pipeline — read a Copernicus tile, reproject it, compute slope, derive viewshed, write a Cloud-Optimized GeoTIFF — runs as one GPU-resident chain rather than seven processes round-tripping through disk. This is the structural reason agent-driven multi-step analysis is viable at machine speed on ForgeGIS where it is not on a sequence of GDAL CLI invocations.

The library is distributed as standard Maven artifacts with no first-party native code, no GDAL native bindings, and no separate native install step. A single shaded jar drops into existing Maven builds. GPU dispatch uses the JOCL OpenCL bindings — JNI bridge bundled inside the JOCL jar, extracted transparently at runtime — and the system OpenCL runtime that ships with GPU drivers. JOCL is the OpenCL backend today, with Vulkan compute as the next backend; the execution model is deliberately vendor-neutral rather than CUDA-locked. Compared to GDAL-based deployments, this eliminates the separate native library installation, multi-source GLIBC compatibility validation, and per-platform shared library distribution that has historically dominated deployment cycles in air-gapped, regulated, and enterprise environments.

ForgeGIS is the substrate that powers Seaglass Foundry's wider product line — including the forthcoming Seaglass Globe™ (a clean-sheet virtual globe in Rust + Vulkan) and the forthcoming ForgeMind™ agent orchestration layer — and is available independently for organizations integrating it into their own platforms, products, and workflows.

Architecture

ForgeGIS is organized around four principles that distinguish it from extension-based approaches to GPU GIS.

GPU-resident pipeline. Data lands on the GPU once and stays there for the duration of a pipeline, moving through successive operations as device buffers rather than being materialized to host memory or disk between stages. Inter-op I/O is the dominant cost in CLI-orchestrated geospatial pipelines; eliminating it is the structural source of the pipeline-level speedups documented in the Performance section.

Pure-Java Maven distribution. ForgeGIS itself contains no first-party native code. GPU dispatch is via the JOCL OpenCL bindings, which package their JNI bridge inside the JOCL jar and extract it transparently at runtime — no separate native install step is required, and no GDAL native bindings or other multi-vendor native libraries enter the build. The deployment artifact is a single shaded jar. JOCL is the OpenCL backend today, with Vulkan compute as the next backend; the execution model is deliberately vendor-neutral rather than CUDA-locked.

Dual-surface API for programmatic and agent use. Every operation in the catalog is a first-class Java method, callable directly from any JVM application. The same operations are exposed through a native MCP server in two complementary patterns: a **pipeline DSL** that composes 133 of the operations into multi-step workflows agents construct dynamically, and **54 curated intent tools** providing typed single-call endpoints for common analytic tasks. The MCP surface totals 62 tools — 54 intent tools, 5 primitive handle and session management tools, and 3 pipeline-escape tools. 218 of 220 operations are MCP-accessible; the two exceptions are Java-only by architectural choice (`Warp` is a higher-level facade redundant with `Reproject` on the MCP surface; `DistanceMatrixCompute` returns a 2D distance matrix that does not fit the MCP raster-handle return contract).

Numerical determinism preserved across surfaces. The MCP server dispatches to the same `ComputeOp` implementations exposed through the direct Java API. The MCP layer is a parameter-marshaling shim, not a parallel implementation — numerical outputs depend on the operation and its parameters, not the invocation surface. The deterministic guarantees scripted workflows depend on remain intact when the same operations are invoked by an AI agent — a precondition for adoption in regulated environments where audit and reproducibility requirements apply.

The library is structured around strict one-directional Maven module dependencies. The compute core (`forgegis-core`, `forgegis-spatial`, category-specific modules) has no dependencies on the I/O layer, the MCP server, or the validation harness. The I/O layer (`forgegis-io`) depends on the compute core and on a small set of vetted geospatial format libraries. The MCP server (`forgegis-mcp`) depends on the catalog and the I/O layer. This separation means consumers who do not need MCP, or who do not need the I/O layer's format support, can depend on subsets of the library without pulling in unnecessary surface area.

Capability Catalog

ForgeGIS provides 220 user-facing operations organized into eleven categories. Each category below summarizes the surface; the complete enumerated catalog appears as Appendix A.

Terrain (14 operations). Slope, aspect, hillshade, curvature, roughness, topographic position index (TPI), terrain ruggedness index (TRI), vector ruggedness measure (VRM), surface area ratio, and related derivatives. The foundation of any DEM-driven analysis.

Bathymetry and Ocean (14 operations). Bathymetric position index, benthic irradiance, depth uncertainty, isopach computation, sound velocity profile (Mackenzie), SOFAR channel identification, and related ocean-specific derivatives. Designed for littoral and underwater acoustic workflows where the raster substrate is bathymetric rather than topographic.

Hydrology (7 operations). Sink filling, flow direction, flow accumulation, stream network extraction, stream order, topographic wetness index (TWI), and watershed delineation. The standard hydrological derivative chain, GPU-resident end to end.

Visibility and RF (8 operations). Viewshed, batched multi-observer viewshed, intervisibility, Fresnel zone computation, insolation, horizon angle, shadow, and sky-view factor. The line-of-sight and propagation surface for defense, telecommunications, and solar analysis use cases.

Spectral and Machine Learning (30 operations). Kriging, inverse distance weighting (IDW), kernel density estimation (KDE), principal component analysis (PCA), minimum noise fraction (MNF), pansharpening, spectral angle mapper, random forest classification, DBSCAN and HDBSCAN clustering, K-means classification, maximum likelihood classification, and related spectral and ML primitives. The category positions ForgeGIS for imagery analytics and AI-driven classification workflows.

Spatial Geometry (80 operations). The largest category. Vector buffering, vector clipping, vector reprojection, geometry validation, geometry simplification, least-cost path, K-nearest-neighbors, and fifteen trajectory analysis operations. Topological relationships are covered to the full DE-9IM surface: the 8 OGC SFS named predicates (Intersects, Disjoint, Within, Contains, Equals, Touches, Crosses, Overlaps), the 3 JTS extensions (Covers, CoveredBy, ContainsProperly), `Relate` for user-supplied DE-9IM mask matching, and the raw 9-character intersection matrix exposed per geometry-type pair through six matrix-computing operations (Point/Linestring/Polygon combinations). The matrix-level exposure means callers can derive any topological relationship a named predicate can't express. Validated against JTS for topology and against analytic oracles for distance and trajectory metrics.

Filtering, Focal, and Morphology (14 operations). Gaussian blur, median filter, majority filter, edge detection, generalized convolution, morphological operations, nibble, sieve, spike removal, no-data filling, and related focal and morphological transforms.

Raster Management, Vectorize, and I/O (39 operations). Reprojection, resampling, clipping, translation, mosaicking, rasterization, polygonization, contour generation, overview construction, band math, map algebra, zonal statistics, and the I/O surface for GeoTIFF, COG, NetCDF, GeoPackage, and standard vector formats.

Routing (7 operations). Traveling salesman, multi-depot TSP, capacitated vehicle routing, time-windowed VRP, service area allocation, distance matrix computation, and nearest-neighbor tour construction. Routing on GPU rather than on a CPU-bound graph engine, with implications for fleet-scale and humanitarian logistics workloads.

Point Cloud (5 operations). Point cloud rasterization, ground filtering, classification, canopy height model derivation, and voxelization. The category is intentionally focused; ForgeGIS positions point cloud as a substrate for raster derivation rather than as an end-to-end LIDAR processing platform.

Temporal (2 operations). Temporal statistics and temporal anomaly detection. The smallest category, intended as a substrate for time-series analysis workflows expressed as repeated invocations of the wider catalog.

66 of the 220 operations have a direct GDAL CLI equivalent; the remaining **154** cover analytic surface that GDAL does not address and that has historically required separate tools — GRASS, SAGA, WhiteboxTools, TauDEM, or specialized commercial libraries — to access.

Performance

All performance numbers in this section are drawn from the May 12, 2026 validation run (identifier `20260512-072709-dff0949e`) on the reference hardware specified in the Deployment section. The full benchmark methodology is documented in Appendix B.

Catalog operations vs. GDAL CLI. Across 143 catalog comparison records — op × raster-scale measurements drawn from the 66 operations with direct GDAL CLI equivalents — ForgeGIS exceeded GDAL by a geometric mean of **6.88×** and a median of **6.51×**, winning 127 of 143 head-to-head comparisons. The catalog-only framing isolates compute performance from format-conversion workloads, where ForgeGIS makes no performance claim and is positioned at parity.

Per-category speedups, ops with GDAL CLI equivalents.

Category	Records	Geomean	Median	Wins
Terrain	5	1.74×	1.57×	5/5
Filtering / Focal / Morphology	6	8.16×	15.95×	5/6
Raster Management / Vectorize / I/O	119	5.29×	5.72×	104/119
Spatial Geometry	13	119.52×	279.06×	13/13

Records are individual op × scale benchmark measurements in the May 2026 run, not unique operations; multiple records typically correspond to one operation timed across several raster scales.

The Spatial Geometry numbers are real but small-N (13 records) and dominated by per-invocation GDAL CLI startup costs on small vector fixtures; the median exceeds the geometric mean (279.06× vs. 119.52×), reflecting a heavy positive tail from a few ops where the CLI startup advantage compounds. They should not be extrapolated to workloads with 10^5 or more features, where the GDAL CLI startup advantage diminishes. Five categories — Bathymetry/Ocean, Hydrology, Visibility/RF, Spectral/ML, and Routing/Point Cloud/Temporal — do not appear in the comparison table above. Individual operations within them (notably the Viewshed operations in Visibility/RF and the gridder/interpolation primitives in Spectral/ML) have GDAL CLI equivalents and are tested elsewhere in the run, but the categories themselves did not produce a sufficient sample of records timed against GDAL at production raster scales to support meaningful per-category statistics in this run.

Multi-stage pipelines. Where ForgeGIS's architecture produces its strongest results is in multi-step pipelines where the GPU-resident data plane eliminates inter-stage I/O entirely:

Pipeline	Output	GDAL warm	ForgeGIS warm	Speedup	ForgeGIS Δheap
TerrainPipeline	4096 × 4096 × 1	1,644.79 ms	113.76 ms	**14.46×**	+204 MB
WarpPipeline	2048 × 2048 × 1	1,353.58 ms	34.05 ms	**39.75×**	+42 MB
AlgebraPipeline	4096 × 4096 × 1	722.03 ms	18.26 ms	**39.54×**	+66 MB

These pipelines represent canonical multi-stage GIS workflows — terrain derivative chains, multi-projection warp sequences, and raster algebra compositions — and are the operational shape in which ForgeGIS's architectural decisions produce their largest measured advantage.

Real-world validation across published datasets. To demonstrate that the synthetic-fixture results generalize, the validation run includes per-op benchmarks across five real-world digital elevation datasets at production resolution:

Dataset	Dimensions	Samples	Median speedup	Max speedup
COP30 — Aleutian Islands (W179°)	3600 × 3600	6	13.76×	26.96×
COP30 — Grand Canyon	3600 × 3600	6	21.93×	37.83×
COP30 — UTM Zone 33N	3127 × 4347	6	20.87×	34.96×
Copernicus 30m — Alps	3600 × 3600	6	21.17×	35.51×
ETOPO — sparse ocean tile	2048 × 2048	3	36.94×	41.35×

The medians span a 13.76×–36.94× range across geographically and morphologically distinct terrain — North Pacific volcanic, high-relief alpine, deeply incised canyon, sparse bathymetric, and projected coordinate system inputs — indicating that the speedup distribution is a property of the architecture and the operations rather than a fixture artifact. The ETOPO sparse-tile result reflects the higher fraction of homogeneous and fill-value cells in open-ocean bathymetry, which interpolation and filtering operations process at lower per-cell cost.

Memory and resource footprint. Pipeline heap usage scales with pipeline depth and output dimensionality, ranging from approximately 42 MB for the WarpPipeline at 2048^2 to 544 MB for a deep hydrology chain (sink-fill, flow direction, flow accumulation, stream network) at 4096^2. The representative TerrainPipeline at 4096^2 uses approximately 204 MB of JVM heap. All measurements were taken with the GPU memory footprint dominated by intermediate device buffers and bounded by the resolution and band count of the input rasters.

What is not claimed. ForgeGIS is positioned at parity with GDAL on format-conversion round-trips (full read-process-write cycles dominated by I/O rather than compute). The catalog-only headline figures exclude these workloads explicitly. ForgeGIS is also positioned at parity with GDAL on standalone single-operation invocations at very small raster scales (64^2 and 256^2), where GDAL CLI process-spawn overhead is comparable to ForgeGIS total execution time and the comparison is not informative about underlying compute performance. The performance claims in this section apply at production raster scales (1024^2 and above) for catalog operations and multi-stage pipelines.

Validation and Coverage

ForgeGIS is engineered for the audit and reproducibility standards of regulated geospatial work. Every shipped operation has unit-test coverage, benchmark coverage, and a documented validation oracle. The May 12, 2026 validation run executes 935 records spanning the full operation catalog across multiple raster scales and fixture classes.

Coverage matrix. The library maintains a coverage spreadsheet tracking each operation against five axes: GDAL or cuSpatial equivalent, MCP exposure, unit test coverage, benchmark coverage, and validation oracle. All 220 user-facing operations have current entries in the matrix. 218 of 220 are MCP-accessible; the two exceptions are Java-only by architectural choice.

Validation oracles. Outputs are validated against domain-appropriate references rather than against a single universal oracle, recognizing that no single reference covers the breadth of the catalog. The convention is documented in Seaglass Foundry's internal spatial computing conventions (available to licensed users and partners) and applied consistently across operations:

JTS (Java Topology Suite) is the primary oracle for Cartesian predicates, topological relationships, and planar geometry operations.

PostGIS is the oracle for spheroidal computations on the WGS84 ellipsoid where Cartesian-only oracles are insufficient.

GDAL CLI is the comparison reference for raster operations with direct GDAL equivalents. Verdicts are scored through a tolerance-tiered framework — bit-exact, floating-point strict, structural-similarity (SSIM), and per-operation continuous-output gates — applied per operation class rather than a single universal threshold.

Analytic numerical fixtures — analytic surfaces, bump targets, ramp gradients, and synthesized inputs with known closed-form outputs — validate operations where no external oracle exists, particularly in terrain, bathymetry, and ML categories.

cuSpatial is recognized as the marketing-comparison peer in the GPU geospatial landscape. By internal convention, cuSpatial is intentionally not used as an oracle: where cuSpatial and JTS disagree (notably on edge-inclusion semantics and certain predicate corner cases), ForgeGIS sides with JTS. Per-op cuSpatial parity tooling is a roadmap item, not a shipped feature.

Validation surface beyond GDAL. 731 of the 935 validation records — 78% of the validation surface — exercise operations with no GDAL CLI equivalent (records whose reference label is not `GDAL`). The principal harness categories accounting for the bulk of this surface, reflecting ForgeGIS's deliberate extension beyond GDAL's analytic reach:

Category area	Records
Terrain	172
Interpolation	92
Zonal and Focal	73
Transform	42
Spectral	42
Algebra	35
Ocean	28
Hydrology	27
ML	22
Viewshed	20
I/O	18
Scalar	18
Vectorize	17
Temporal	15
Styling	8
Routing	4

The category names in the validation distribution above use the bench harness's finer-grained analytical taxonomy and do not map one-to-one to the catalog category names used elsewhere in this brief; several validation categories typically correspond to operations within a single catalog category.

This is the surface area on which ForgeGIS makes capability claims rather than performance claims — the operations that exist in ForgeGIS because organizations need them and that have historically required specialized tools (GRASS, SAGA, WhiteboxTools, TauDEM) outside the GDAL toolchain.

Cross-surface determinism. The MCP server dispatches to the same `ComputeOp` implementations exposed through the direct Java API. The MCP layer is a parameter-marshaling shim, not a parallel implementation — numerical outputs depend on the operation and its parameters, not the invocation surface.

Randomness and reproducibility. Most operations in the catalog are deterministic by construction. The two operations in this release that involve randomness (`KMeansClassifyOp` and `RandomForestOp`) run with a fixed deterministic default seed under MCP invocation, so MCP runs are reproducible without per-call configuration. The direct Java API additionally exposes the seed as a constructor parameter for callers that need to vary it across runs. The validation harness verifies determinism by re-executing each operation in the same JVM and comparing outputs byte-for-byte; the May 2026 run reports the full record set passing this gate.

What this validation framework is designed to support. The framework is designed so that an evaluator can ask three independent questions and receive defensible answers: *Is the operation correct?* (answered by oracle match), *Does the operation produce the same result when invoked by an agent as when invoked by a script?* (answered by surface-equivalence by construction), and *Has the validation surface kept pace with the catalog?* (answered by the coverage matrix).

Deployment and Integration

Reference hardware. All performance figures in this brief were measured on commodity laptop hardware: Acer Nitro ANV16-72, Intel® Core™ 7 240H (10C/16T), 15.7 GB system RAM, NVIDIA® GeForce® RTX™ 5070 Laptop GPU (8150 MB VRAM, 36 compute units), running Microsoft Windows® 11 Home (build 26200), OpenJDK Corretto 26.0.0.35.2, GDAL 3.12.1, NVIDIA driver 573.22, OpenCL 3.0 (CUDA 12.8.97). The reference platform is deliberately a single laptop — ForgeGIS does not require server-class infrastructure to deliver the performance characteristics documented in this brief.

Software requirements. Java 17 or later. A GPU with OpenCL 1.2 or later support; NVIDIA, Intel, and AMD platforms are all targets. Apache Maven for build integration. No GDAL installation required at runtime — the GDAL CLI dependency exists only in the benchmark harness for comparative measurement.

Maven coordinates. The library is distributed as standard Maven artifacts under the `com.seaglassfoundry` group identifier. The core compute, I/O, and MCP server modules are independently consumable; applications that do not require the MCP server or the full I/O surface can depend on the subset they need.

MCP server deployment. The MCP server ships as a single shaded executable jar, launched as a standalone process and communicating over JSON-RPC on stdio per the MCP specification. First-deploy cold start (no OS file cache): approximately 3 seconds to first usable response. Process-restart cold start (typical operational case with warm OS cache): approximately 800 ms median to first `tools/list` response, returning all 62 MCP tools. The `initialize` → `tools/list` gap is approximately 100 ms; the tools list is constructed once on first build and cached for subsequent calls.

Air-gapped and regulated deployment. Because ForgeGIS contains no first-party native code and no GDAL native bindings, deployment to air-gapped, classified, or otherwise restricted environments does not require coordinating separate native binary distribution, multi-source GLIBC validation, or shared-library signing for ForgeGIS itself. The deployment artifact is the same Maven jar (or shaded jar, for standalone MCP server) that runs in unrestricted environments. The JOCL OpenCL bindings ship their JNI bridge bundled inside the JOCL jar, extracted at runtime; the system OpenCL runtime required for GPU dispatch is standard system software bundled with GPU drivers. Required external components — JDK, GPU driver, OpenCL runtime — all have established procurement and approval paths in regulated environments.

Licensing and Contact

Licensing. ForgeGIS license terms are being finalized and will be published prior to general availability. Organizations with specific licensing requirements are invited to contact Seaglass Foundry directly.

Partnership and distribution. Seaglass Foundry is seeking strategic partners and distributors positioned to bring ForgeGIS to defense, intelligence, climate, and enterprise markets at scale. Inquiries from organizations with relevant distribution, integration, or capital-deployment capacity are welcome.

Contact. Seaglass Foundry · rich@seaglassfoundry.com · seaglassfoundry.com

About Seaglass Foundry

Seaglass Foundry LLC is an independent geospatial software company based in Panama City, Florida, building the compute substrate for AI-native geospatial analysis. Its product line includes ForgeGIS (GPU-accelerated geospatial library and MCP server), Seaglass Globe (Rust + Vulkan virtual globe, forthcoming), ForgeMind (LLM-agnostic agent orchestration layer, forthcoming), and SwingToPDF™ (vector PDF export for Java Swing applications, in production).

Appendix A — Complete Operation Catalog

The 220 user-facing operations in ForgeGIS, organized by category. The *API Surface* column indicates which interfaces expose each operation: **Java** for the direct Java API, **MCP** for an intent tool on the MCP server, **DSL** for inclusion in the MCP pipeline DSL composition set. *GDAL CLI Equivalent* lists the direct command-line analogue where one exists; — indicates either no GDAL CLI equivalent or that the GDAL alternative is not a direct substitute.

Terrain (14)

Operation	GDAL CLI Equivalent	API Surface
`Aspect`	gdal raster aspect	Java + MCP + DSL
`ConvergenceIndex`	—	Java + MCP + DSL
`Curvature`	—	Java + MCP + DSL
`HeatLoadIndex`	—	Java + MCP + DSL

Operation	GDAL CLI Equivalent	API Surface
`Hillshade`	gdal raster hillshade	Java + MCP + DSL
`ReefComplexityIndex`	—	Java + MCP + DSL
`RelativeElevation`	—	Java + MCP + DSL
`Roughness`	gdal raster roughness	Java + MCP + DSL
`Rugosity`	—	Java + MCP + DSL
`Slope`	gdal raster slope	Java + MCP + DSL
`SurfaceAreaRatio`	—	Java + MCP + DSL
`Tpi`	gdal raster tpi	Java + MCP + DSL
`Tri`	gdal raster tri	Java + MCP + DSL
`Vrm`	—	Java + MCP + DSL

Bathymetry and Ocean (14)

Operation	GDAL CLI Equivalent	API Surface
`BackscatterClassify`	—	Java + MCP + DSL
`BathymetricGradientCurrent`	—	Java + MCP + DSL
`BathymetricPositionIndex`	—	Java + MCP + DSL
`BenthicIrradiance`	—	Java + MCP + DSL
`DepthDerivedWaveExposure`	—	Java + MCP + DSL
`DepthExcess`	—	Java + MCP
`DepthUncertainty`	—	Java + MCP + DSL
`DredgeVolume`	—	Java + MCP + DSL
`Isopach`	—	Java + MCP + DSL
`SofarChannel`	—	Java + MCP
`SoundVelocityProfile`	—	Java + MCP + DSL
`SurfaceDuct`	—	Java + MCP
`SvpInterpolation3D`	—	Java + MCP
`TidalCurrent`	—	Java + MCP + DSL

Hydrology (7)

Operation	GDAL CLI Equivalent	API Surface
`FlowAccumulation`	—	Java + MCP + DSL
`FlowDirection`	—	Java + MCP + DSL

Operation	GDAL CLI Equivalent	API Surface
`SinkFill`	—	Java + MCP + DSL
`StreamNetwork`	—	Java + MCP + DSL
`StreamOrder`	—	Java + MCP + DSL
`Twi`	—	Java + MCP + DSL
`Watershed`	—	Java + MCP + DSL

Visibility and RF (8)

Operation	GDAL CLI Equivalent	API Surface
`BatchViewshed`	gdal raster viewshed (cumulative mode)	Java + MCP + DSL
`FresnelZone`	—	Java + MCP + DSL
`HorizonAngle`	—	Java + MCP + DSL
`Insolation`	—	Java + MCP + DSL
`Intervisibility`	—	Java + MCP + DSL
`Shadow`	—	Java + MCP + DSL
`SkyViewFactor`	—	Java + MCP + DSL
`Viewshed`	gdal raster viewshed	Java + MCP + DSL

Spectral and Machine Learning (30)

Operation	GDAL CLI Equivalent	API Surface
`AverageDistanceGridder`	gdal vector grid average-distance	Java + MCP + DSL
`AverageDistancePointsGridder`	gdal vector grid average-distance-points	Java + MCP + DSL
`AverageGridder`	gdal vector grid average	Java + MCP + DSL
`BenthicTerrainClassify`	—	Java + MCP + DSL
`CountGridder`	gdal vector grid count	Java + MCP + DSL
`Dbscan`	—	Java + MCP + DSL
`Hdbscan`	—	Java + MCP + DSL
`Idw`	gdal vector grid invdist / invdistnn	Java + MCP + DSL
`KMeansClassify`	—	Java + MCP + DSL
`Kde`	—	Java + MCP + DSL
`Kriging`	—	Java + MCP + DSL
`LandformClassify`	—	Java + MCP + DSL
`LinearInterpolation`	gdal vector grid linear	Java + MCP + DSL

Operation	GDAL CLI Equivalent	API Surface
`MaxLikelihood`	—	Java + MCP + DSL
`MaximumGridder`	gdal vector grid maximum	Java + MCP + DSL
`MinimumGridder`	gdal vector grid minimum	Java + MCP + DSL
`Mnf`	—	Java + MCP + DSL
`NaturalNeighbor`	—	Java + MCP + DSL
`NearestNeighborInterpolation`	gdal vector grid nearest	Java + MCP + DSL
`Pansharpen`	—	Java + MCP + DSL
`Pca`	—	Java + MCP + DSL
`RandomForest`	—	Java + MCP + DSL
`RangeGridder`	gdal vector grid range	Java + MCP + DSL
`Segment`	—	Java + MCP + DSL
`SpectralAngleMapper`	—	Java + MCP + DSL
`SpectralUnmixing`	—	Java + MCP + DSL
`Spline`	—	Java + MCP + DSL
`TasseledCap`	—	Java + MCP + DSL
`Tin`	—	Java + MCP + DSL
`TrendSurface`	—	Java + MCP + DSL

Spatial Geometry (80)

Operation	GDAL CLI Equivalent	API Surface
`BallQuery`	—	Java + MCP
`BboxIntersect`	—	Java + MCP
`Buffer`	—	Java + MCP + DSL
`BufferVector`	gdal vector geom buffer	Java + MCP + DSL
`CartesianDistance`	—	Java + MCP
`ConcatVector`	gdal vector concat	Java + MCP + DSL
`Contains`	—	Java + MCP
`ContainsProperly`	—	Java + MCP
`Corridor`	—	Java + MCP + DSL
`CostDistance`	—	Java + MCP + DSL
`CoveredBy`	—	Java + MCP
`Covers`	—	Java + MCP

Operation	GDAL CLI Equivalent	API Surface
`Crosses`	—	Java + MCP
`De9imLinestringLinestring`	—	Java + MCP
`De9imLinestringPolygon`	—	Java + MCP
`De9imPointLinestring`	—	Java + MCP
`De9imPointPoint`	—	Java + MCP
`De9imPointPolygon`	—	Java + MCP
`De9imPolygonPolygon`	—	Java + MCP
`DeriveTrajectories`	—	Java + MCP
`DirectedHausdorffDistance`	—	Java + MCP
`Disjoint`	—	Java + MCP
`EditVector`	gdal vector edit	Java + MCP + DSL
`Equals`	—	Java + MCP
`EuclideanAllocation`	—	Java + MCP + DSL
`EuclideanDistance`	—	Java + MCP + DSL
`FilterVector`	gdal vector filter	Java + MCP + DSL
`HausdorffDistance`	—	Java + MCP
`HaversineDistance`	—	Java + MCP
`Intersects`	—	Java + MCP
`JoinQuadtreeAndBoundingBoxes`	—	Java + MCP
`KNearestNeighbors`	—	Java + MCP
`LeastCostPath`	—	Java + MCP + DSL
`LinestringLinestringDistance`	—	Java + MCP
`LinestringLinestringIntersection`	—	Java + MCP
`LinestringPolygonDistance`	—	Java + MCP
`MakeValidVector`	gdal vector geom make-valid	Java + MCP + DSL
`ManhattanDistance`	—	Java + MCP
`NearestLinestring`	—	Java + MCP
`NearestPolygon`	—	Java + MCP
`Overlaps`	—	Java + MCP
`PairwiseWithinDistance`	—	Java + MCP
`PointInPolygonQuadtree`	—	Java + MCP
`PointInPolygonScan`	—	Java + MCP

Operation	GDAL CLI Equivalent	API Surface
`PointLinestringDistance`	—	Java + MCP
`PointPolygonDistance`	—	Java + MCP
`PointsInSpatialWindow`	—	Java + MCP
`PointsWithinDistanceOfLinestrings`	—	Java + MCP
`PointsWithinDistanceOfPolygons`	—	Java + MCP
`PolygonPolygonDistance`	—	Java + MCP
`PredicatePipeline`	—	Java + MCP
`QuadtreeOnPoints`	—	Java + MCP
`Relate`	—	Java + MCP
`SegmentizeVector`	gdal vector geom segmentize	Java + MCP + DSL
`SelectVector`	gdal vector select	Java + MCP + DSL
`SetTypeVector`	gdal vector geom set-type	Java + MCP + DSL
`SimplifyVector`	gdal vector geom simplify	Java + MCP + DSL
`SpatialJoinAttribute`	—	Java + MCP
`SspdDistance`	—	Java + MCP
`SwapXyVector`	gdal vector geom swap-xy	Java + MCP + DSL
`Touches`	—	Java + MCP
`TrajectoryAverageSpeed`	—	Java + MCP
`TrajectoryDuration`	—	Java + MCP
`TrajectoryEnvelope`	—	Java + MCP
`TrajectoryPipeline`	—	Java + MCP
`TrajectoryResample`	—	Java + MCP
`TrajectorySegmentAcceleration`	—	Java + MCP
`TrajectorySegmentBearing`	—	Java + MCP
`TrajectorySegmentLengths`	—	Java + MCP
`TrajectorySegmentSpeed`	—	Java + MCP
`TrajectorySimilarityHausdorff`	—	Java + MCP
`TrajectorySimilaritySspd`	—	Java + MCP
`TrajectorySimplifyDP`	—	Java + MCP
`TrajectoryStopDetection`	—	Java + MCP
`TrajectoryTotalDistance`	—	Java + MCP
`VectorClip`	gdal vector clip	Java + MCP + DSL

Operation	GDAL CLI Equivalent	API Surface
`VectorInfo`	gdal vector info	Java + MCP + DSL
`VectorReproject`	gdal vector reproject	Java + MCP + DSL
`VectorSql`	gdal vector sql	Java + MCP + DSL
`Within`	—	Java + MCP

Filtering, Focal, and Morphology (14)

Operation	GDAL CLI Equivalent	API Surface
`AnisotropicDiffusion`	—	Java + MCP + DSL
`CleanCollar`	gdal raster clean-collar	Java + MCP + DSL
`ColorRelief`	gdal raster color-map	Java + MCP + DSL
`Convolve`	—	Java + MCP + DSL
`EdgeDetect`	—	Java + MCP + DSL
`FillNodata`	gdal raster fill-nodata	Java + MCP + DSL
`Focal`	—	Java + MCP + DSL
`GaussianBlur`	—	Java + MCP + DSL
`MajorityFilter`	—	Java + MCP + DSL
`MedianFilter`	—	Java + MCP + DSL
`Morph`	—	Java + MCP + DSL
`Nibble`	—	Java + MCP + DSL
`SieveFilter`	gdal raster sieve	Java + MCP + DSL
`SpikeFilter`	—	Java + MCP + DSL

Raster Management, Vectorize, and I/O (39)

Operation	GDAL CLI Equivalent	API Surface
`BandConcat`	gdal raster stack	Java + MCP + DSL
`BandMath`	gdal raster calc	Java + MCP + DSL
`BandSelect`	gdal raster select	Java + MCP + DSL
`BandStatistics`	gdal raster info -stats (composed)	Java + MCP + DSL
`Blend`	gdal raster blend	Java + MCP + DSL
`ChangeDetect`	—	Java + MCP + DSL
`Clip`	gdal raster clip	Java + MCP + DSL
`ClipByMask`	gdal raster clip	Java + MCP + DSL

Operation	GDAL CLI Equivalent	API Surface
`Compare`	gdal raster compare	Java + MCP
`Contour`	gdal raster contour	Java + MCP + DSL
`ConvertVerticalDatum`	—	Java + MCP + DSL
`CreateRaster`	gdal raster create	Java + MCP
`CutFillVolume`	—	Java + MCP + DSL
`Cutline`	gdal raster clip	Java + MCP + DSL
`DataTypeConvert`	gdal raster set-type	Java + MCP + DSL
`Footprint`	gdal raster footprint	Java + MCP + DSL
`Glcm`	—	Java + MCP + DSL
`Histogram`	gdal raster info -hist (composed)	Java + MCP + DSL
`MapAlgebra`	gdal raster calc (legacy gdal_calc.py)	Java + MCP + DSL
`MetadataEdit`	gdal raster edit	Java + MCP + DSL
`Mosaic`	gdal raster mosaic	Java + MCP + DSL
`MultiMosaic`	gdal raster mosaic	Java + MCP
`Overview`	gdal raster overview add	Java + MCP
`PixelInfo`	gdal raster pixel-info	Java + MCP + DSL
`PolygonToMask`	—	Java + MCP + DSL
`Polygonize`	gdal raster polygonize	Java + MCP + DSL
`RasterTileIndex`	gdal raster index	Java + MCP
`Rasterize`	gdal vector rasterize	Java + MCP
`Reclassify`	gdal raster reclassify	Java + MCP + DSL
`Reproject`	gdal raster reproject	Java + MCP + DSL
`ReprojectForTessellation`	—	Java + MCP
`Resample`	gdal raster resize	Java + MCP + DSL
`Rescale`	gdal raster scale	Java + MCP + DSL
`Retile`	—	Java + MCP
`RgbEncode`	—	Java + MCP + DSL
`Translate`	gdal raster convert	Java + MCP + DSL
`Unscale`	gdal raster unscale	Java + MCP + DSL
`Warp`	gdalwarp (broader facade — CRS-only via gdal raster reproject)	Java only
`ZonalStats`	—	Java + MCP + DSL

Routing (7)

Operation	GDAL CLI Equivalent	API Surface
`DistanceMatrixCompute`	—	Java only
`MultiDepotTSP`	—	Java + MCP
`NearestNeighborTour`	—	Java + MCP
`ServiceAreaAllocation`	—	Java + MCP
`TimeWindowedVRP`	—	Java + MCP
`TravelingSalesman`	—	Java + MCP
`VehicleRouting`	—	Java + MCP

Point Cloud (5)

Operation	GDAL CLI Equivalent	API Surface
`CanopyHeightModel`	—	Java + MCP
`GroundFilter`	—	Java + MCP
`PointCloudClassify`	—	Java + MCP
`PointCloudToRaster`	—	Java + MCP
`Voxelize`	—	Java + MCP

Temporal (2)

Operation	GDAL CLI Equivalent	API Surface
`TemporalAnomaly`	—	Java + MCP
`TemporalStats`	—	Java + MCP

Appendix B — Benchmark Methodology

Run identifier. `20260512-072709-dff0949e` (May 12, 2026, 07:27 UTC).

Reference hardware. Acer Nitro ANV16-72; Intel Core 7 240H (10C/16T); 15.7 GB RAM; NVIDIA GeForce RTX 5070 Laptop GPU (8150 MB VRAM, 36 compute units); Windows 11 Home build 26200; OpenJDK Corretto 26.0.0.35.2; NVIDIA driver 573.22; OpenCL 3.0 (CUDA 12.8.97); JVM heap max 4096 MB. The bench harness used GDAL 3.12.1 as the comparison reference (the system-wide `gdalinfo --version` reports 3.11.4 in some environments; the harness-bundled 3.12.1 is authoritative for benchmark numbers).

Run scope. 935 total benchmark records across the 220-operation catalog, organized into the per-op category benchmarks, the multi-stage pipeline benchmarks, and the real-world dataset benchmarks reported in the

Performance section. 731 of 935 records (78%) test operations with no GDAL CLI equivalent (records with reference labels other than `GDAL`) and use the validation-oracle convention documented in `docs/spatial-compute-conventions.md`.

Timing methodology. Each measurement runs six warm iterations per operation per scale; the head-to-head and per-category figures in this brief report warm-minimum times across those iterations (excluding the cold first invocation, which captures JIT warmup and OS-cache effects). Median and geometric-mean statistics are computed across the per-op series. Cold-start figures, where reported, are explicitly identified.

Raster scale strategy. Headline performance figures are drawn from production raster scales of 1024^2 and 4096^2, where compute time dominates and CLI process-spawn overhead is a small fraction of total measured time. Scales of 64^2 and 256^2 are included in the full run for completeness and to disarm cherry-picking criticism but are excluded from headline materials because at those scales GDAL CLI process-spawn overhead can dominate ForgeGIS total execution time, making the comparison non-informative about underlying compute performance.

Excluded categories of records. 61 of the 204 GDAL CLI comparison records are excluded from the catalog-only headline figures. 52 are I/O round-trip records — full read-process-write cycles dominated by format conversion rather than compute — on which ForgeGIS is positioned at parity with GDAL. The remaining 9 are format-management or write-path records where a GDAL CLI comparison is not informative about underlying compute performance. The catalog-only headline figures (6.88× geomean / 6.51× median / 127 wins across 143 catalog records) exclude these 61 records explicitly. The full-run figures including I/O round-trips and format-management records (3.48× geomean / 2.94× median / 148 wins across 204 records) are auditable from the underlying record set.

Pipeline construction. The multi-stage pipeline benchmarks (TerrainPipeline, WarpPipeline, AlgebraPipeline) construct GDAL CLI chains using temporary GeoTIFF files between stages, which is the standard way these workflows are orchestrated outside an in-process compute library. ForgeGIS executes the same conceptual chain GPU-resident with no intermediate disk writes. This is a fair comparison of the operational shapes the two systems are designed for, not an apples-to-oranges comparison of in-process vs. CLI execution.

Real-world datasets. The five datasets used for the real-world validation table (COP30 Aleutian Islands W179°, COP30 Grand Canyon, COP30 UTM33N, Copernicus 30m Alps, ETOPO sparse tile) were chosen to span geographically and morphologically distinct terrain (North Pacific volcanic, high-relief alpine, deeply incised canyon, projected coordinate inputs, sparse bathymetric ocean). All datasets are publicly available from their respective sources.

Verdicts. Operations with GDAL CLI equivalents are verdicted by structural similarity (SSIM) on output rasters with appropriate tolerances per operation class. Operations without GDAL CLI equivalents are verdicted against the validation oracles described in the Validation and Coverage section. The full per-op verdict record is preserved in the run JSONL output.

Trademarks. ForgeGIS, Seaglass Foundry, Seaglass Globe, ForgeMind, and SwingToPDF are trademarks of Seaglass Foundry LLC. GDAL, JTS, PostGIS, OpenCL, Vulkan, NVIDIA, GeForce, RTX, CUDA, cuSpatial,