Spatial Lambdas

VoxelSpace's **spatial lambdas** are serverless functions that execute custom computations on your volumetricdata.

Introduction

VoxelSpace's **spatial lambdas** are serverless functions that execute custom computations on your volumetricdata. They run near your data on highly parallel infrastructure, allowing you to process trillions of voxels inminutes. This guide explains what spatial lambdas are, when to use them, how to write and execute them, and bestpractices for efficient processing.

What are Spatial Lambdas?

Spatial lambdas are stateless cloud functions written in languages such as **Python** or **.NET**. They take voxel gridsas input, perform a computation, and return a result or a modified dataset. Examples include:

- **Volume calculations** and statistical summaries
- **Filtering voxels** by property values
- **Creating derived datasets** from existing data
- **Classifying voxels** with machine learning models

Because the lambdas execute server-side, they scale to trillions of voxels through massive parallelization.

When to Use a Lambda

Use a spatial lambda when you need to:

- Compute metrics such as t**otal volume tonnage average density**, or other summarystatistics
- **Filter** voxels based on property thresholds (e.g., density > 2.5) or remove empty voxels
- **Resample** data to a coarser or finer resolution
- **Classify** voxels with a machine learning model (e.g., ore vs. waste, rock type, or vegetationclass)
- Generate **reports** of computed metrics for compliance or engineering studies

Authoring a Lambda

Spatial lambdas are typically composed of three main components:

1. Input Definition

Specify the dataset(s) and region of interest (bounding box or selection) to process.

2. Compute Function

Write code that iterates through voxels and applies your logic. For example, summing the ore_grade property forvoxels above a grade threshold.

3. Output

Return a summary value (e.g., total_volume) or create a new dataset (e.g., filtered voxels). Lambdas can alsoattach files or emit logs during execution.

Sample Python Lambda

Here's a basic example that sums the volume of high-grade voxels:

```
def lambda_handler (context, voxel_reader, voxel_writer):
  total_volume = 0.0
  for voxel in voxel_reader:
    if voxel.properties.get('grade',0.0) &gt;= 0.5:
      total_volume += voxel.size
  return {'total_volume': total_volume}
```

Code Explanation

voxel_reader**:** Provides voxel objects with property dictionaries and geometry

voxel.properties.get('grade', 0.0)**:** Safely retrieves the grade property with a default valueof 0.0

voxel.size**:** Represents the volume of a voxel (e.g., 1 m^3)

Return value: Dictionary with the computed total volume that will appear in reports

Running a Lambda

Follow these steps to execute a spatial lambda:

Step 1:**Prepare Your Dataset**

Ensure your data is processed into an **indexed dataset** or **voxel grid**. Lambdas operate on indexed datasets ratherthan raw uploads.

Step 2:**Select or Upload a Lambda**

Navigate to your dataset's actions menu

Choose **Run Lambda**

Select a pre-built lambda (e.g., volume calculation, surface extraction) OR upload your owncode file

– – –

Step 3:Configure Parameters

Define the execution parameters:

Region of interest: Full dataset or specific bounding box

Property filters: Set thresholds (e.g., grade ? 0.5)

Output options: Specify whether to generate new datasets

User inputs: Provide any custom parameters your lambda requires

Step 4:Launch the Job

Start the lambda execution:

The task appears in **Pending Tasks**

On the free tier, only one job (processing or lambda) can run at a time

Additional jobs queue until the current job finishes

Step 5:Review Results

When complete, results appear in:

Reports section: Summary values and statistics

Outputs section: New datasets generated by the lambda

Best Practices

Performance Optimization

Limit the region: Restrict the lambda to the smallest area necessary to reduce compute costand time

Efficient algorithms: Use vectorized operations where possible for better performance

Memory management: Be mindful of memory usage when processing large datasets

Development Workflow

Test on a sample: Validate your code on a small subset before running it on the full dataset

Incremental development: Start with simple logic and gradually add complexity

Error handling: Include proper error handling for robust execution

Resource Management

Reuse templates: Start with VoxelSpace's built-in lambdas for common tasks; modify themrather than writing from scratch

Queue awareness: Plan your processing and lambda jobs to avoid long queues, especially onthe free tier where concurrency is limited

Monitor usage: Track your compute usage to stay within plan limits

Debugging and Validation

Use logs: Include log statements in your lambda to help diagnose issues and verifyintermediate results

Validate inputs: Check that your data has the expected properties before processing

Test edge cases: Consider what happens with empty regions, missing properties, or extremevalues

Advanced Lambda Example

Here's a more sophisticated lambda that performs statistical analysis:

```
def statistical_analysis_lambda(context, voxel_reader, voxel_writer):
  grades = []
  volumes = []
  for voxel in voxel_reader:
    grade = voxel.properties.get('grade',0.0)
    if grade &gt; 0: # Only include non-zero grades
      grades.append(grade)
      volumes.append(voxel.size)
  if not grades:
    return {'error':'No valid grade values found'}

  # Calculate statistics
  total_volume = sum(volumes)
  weighted_average_grade = sum(g * v for g, v in zip(grades, volumes)) / total_volume

  return {
    'total_volume': total_volume,
    'voxel_count':len(grades),
    'average_grade':sum(grades) / len(grades),
    'weighted_average_grade': weighted_average_grade,
    'min_grade': min(grades),
    'max_grade': max(grades)
  }
```

Troubleshooting Common Issues

Performance Problems

- **Large memory usage:** Process voxels in batches rather than loading all into memory
- **Slow execution:** Optimize algorithms and reduce unnecessary computations
- **Timeout errors:** Break large jobs into smaller regions

Data Issues

- **Missing properties:** Always use the `.get()` method with default values
- **Unexpected data types:** Validate and convert data types as needed
- **Empty regions:** Check for zero voxel counts before performing calculations

Runtime Errors

- **Import errors:** Ensure all required libraries are available in the runtime environment
- **Syntax errors:** Test your code locally before uploading
- **Logic errors:** Use logging extensively to trace execution flow

Conclusion

Spatial lambdas bring high-performance analytics directly to your data, enabling custom computations at massivescale. By understanding how to author, configure, and run these serverless functions, you can:

- **Extract deeper insights** from voxel models
- **Automate complex spatial analyses**
- **Process trillions of voxels** efficiently
- **Create custom workflows** tailored to your specific needs

The power of spatial lambdas lies in their ability to execute near your data, eliminating the need to transfermassive datasets and enabling real-time analysis at unprecedented scales.

Next Steps

To advance your spatial lambda expertise:

1. **Explore built-in templates** available in the VoxelSpace platform
2. **Experiment with different processing techniques** and algorithms
3. **Study advanced examples** in the developer documentation
4. **Consider upgrading your plan** for increased compute resources and concurrent processing
5. **Join the VoxelSpace community** to share techniques and learn from other users

For comprehensive developer documentation and advanced tutorials, consult the full manuals within VoxelSpace once you have platform access.