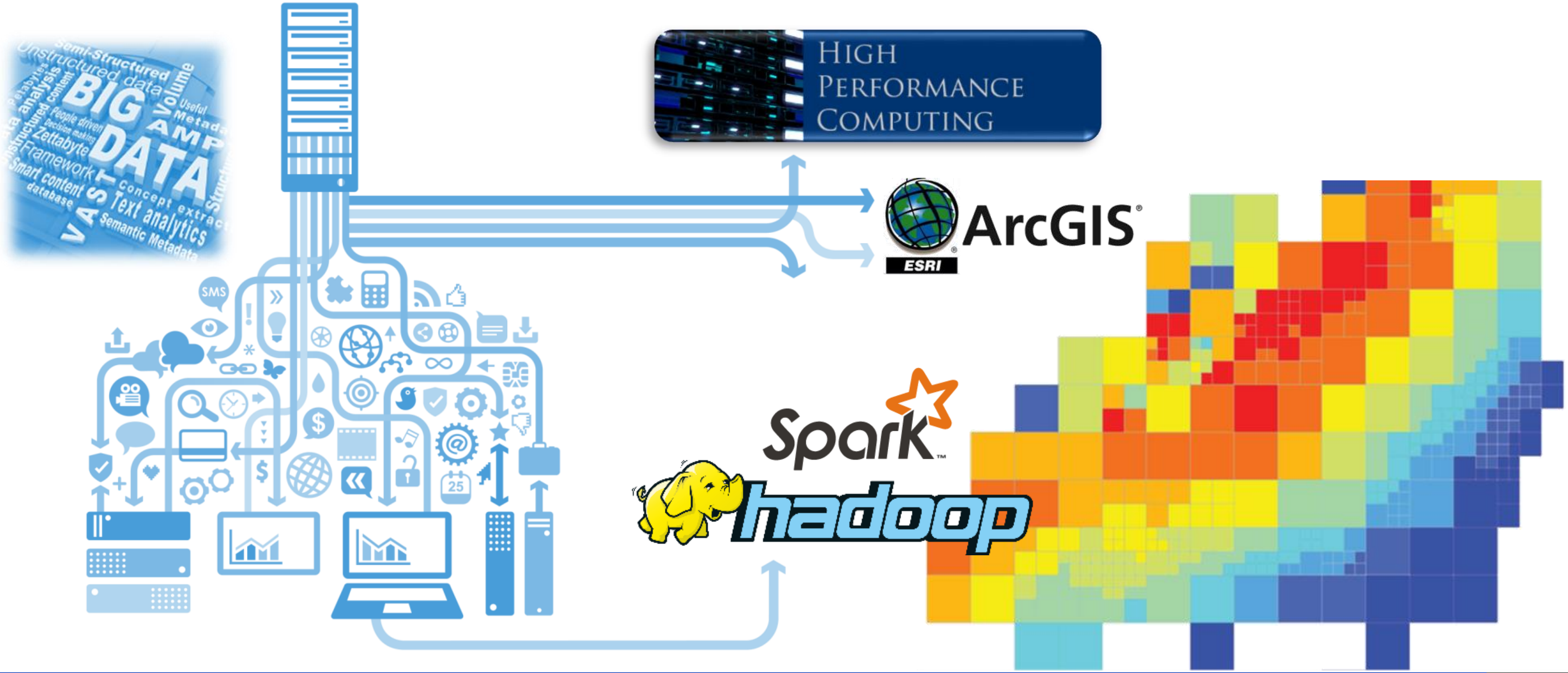
Leveraging Big Data Computing through EDX for Advanced Energy R&D & Analytics

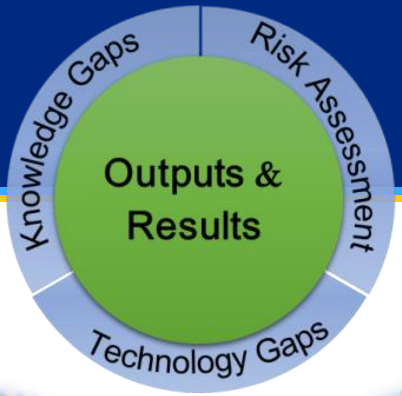


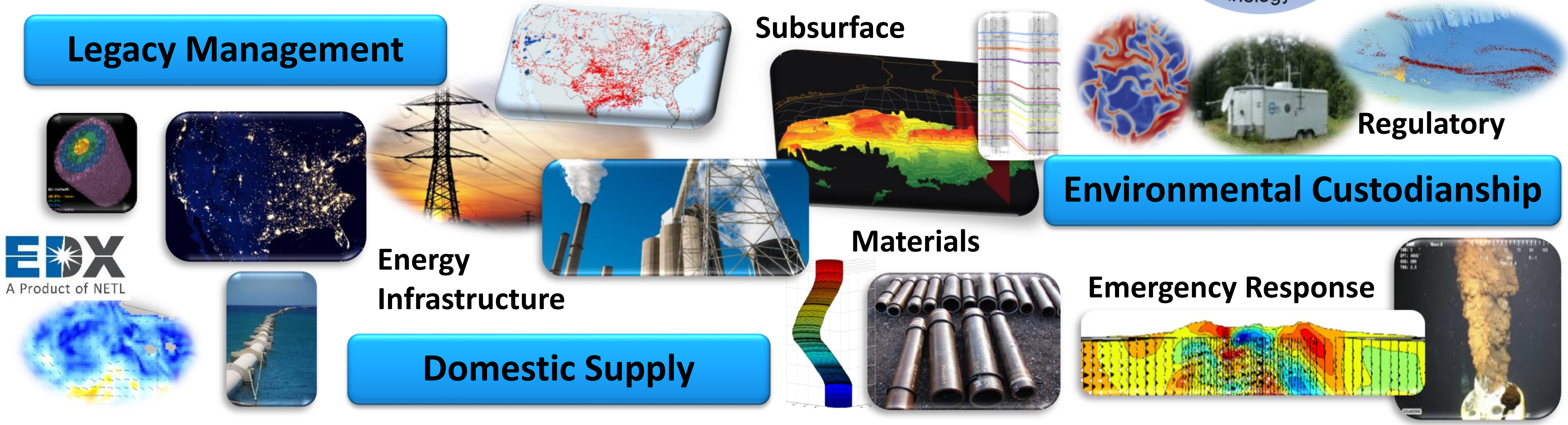
Vic Baker, Kelly Rose, Jennifer Bauer, Dave Rager

National Energy Technology Laboratory, U.S. Department of Energy

8/18/16

Data from & for energy R&D

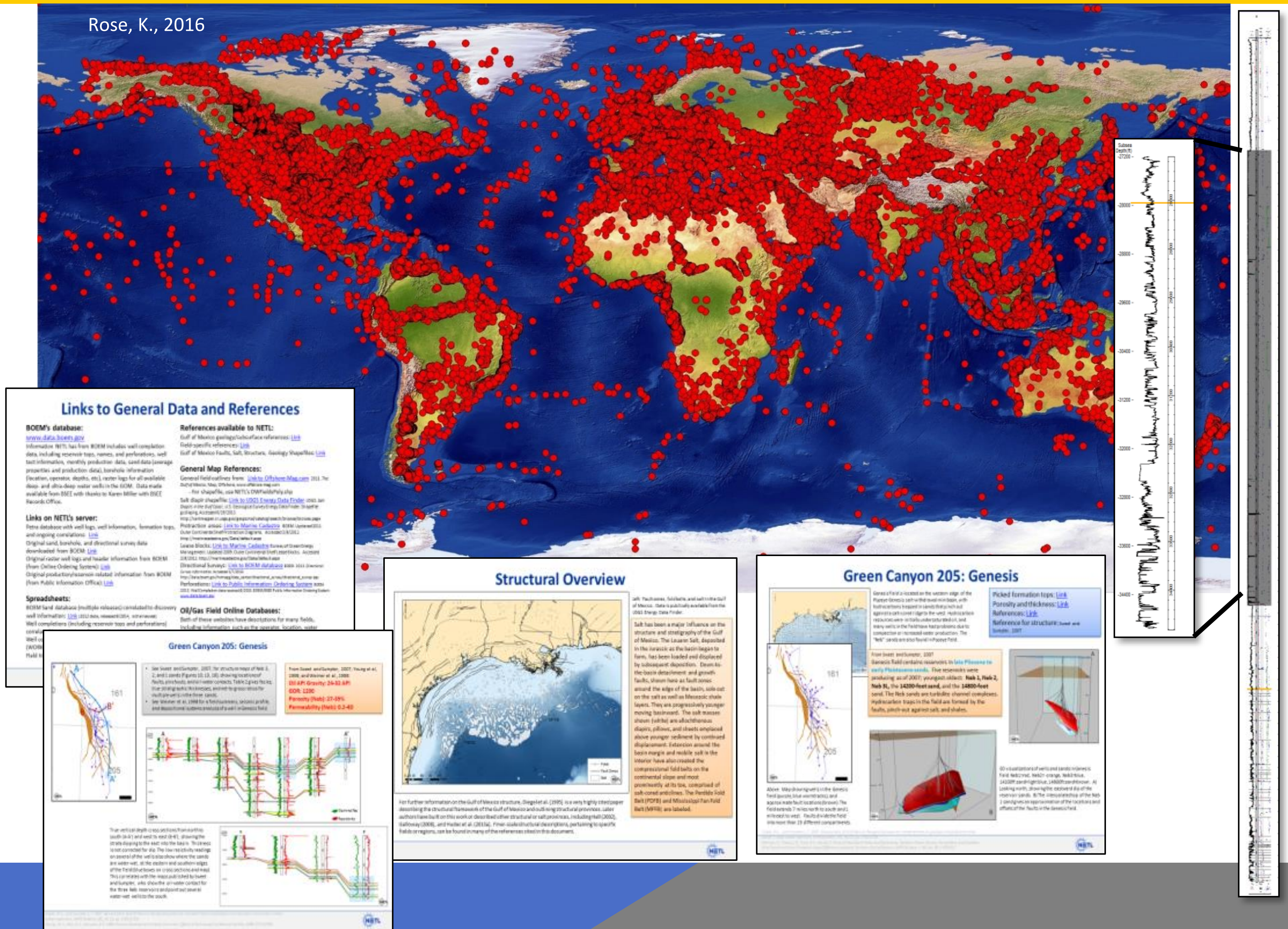




Spanning the **subsurface to atmosphere, engineered & natural systems**

Common Energy Data Challenges

- ***Finding & accessing*** authoritative, appropriate data
- ***Multiple scales and sources***
- Often ***indirect sources***
- ***Discontinuous*** data
- ***Missing*** data
- Numerous forms of ***uncertainty***
- Multi-component data
- ***Cost*** of data preservation
- Challenge of ***big data***
- Historical, ***non digital*** datasets
- Structured & ***unstructured***
- Representing ***numerous systems***
- Spanning ***numerous sources***



Finding & accessing data

NETL's Energy Data Exchange (EDX) provides an ***innovative*** solution for data-driven efforts offering:

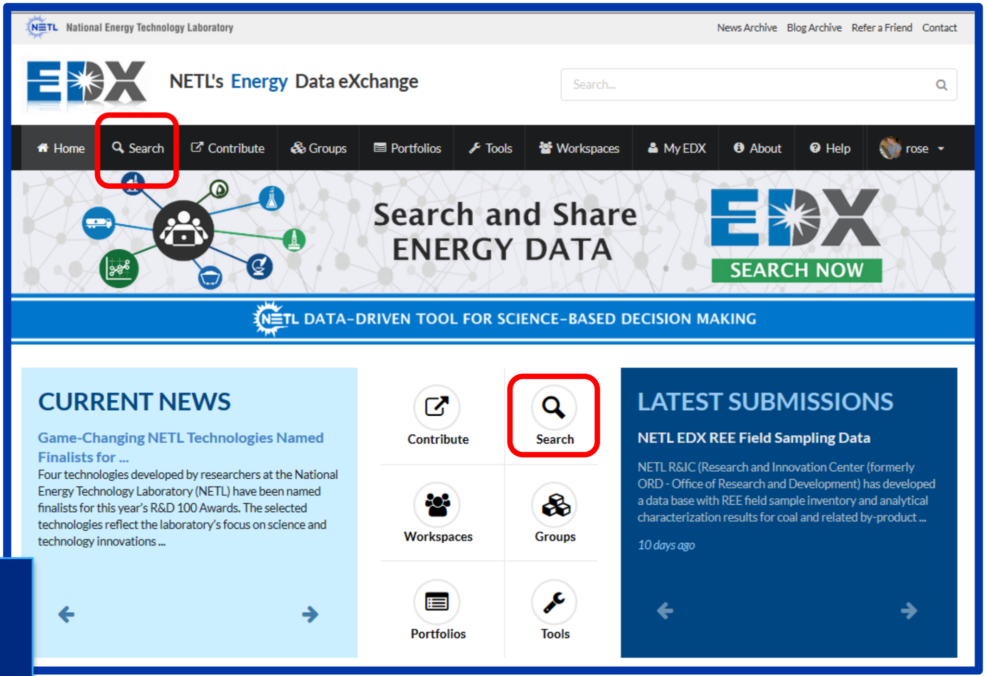
- A secure, online ***coordination and collaboration platform*** supporting energy research, knowledge transfer and data ***discovery*** needs
- Enduring and reliable ***access*** to historic and current R&D ***data, data driven products, and tools***
- Offers both ***public*** and ***secure, private*** functionalities

EDX serves as a liaison between data resources and future needs

Public Side
Enables knowledge transfer, data preservation, reuse & discovery

Secure/Private Side
Supports research development, collaboration, & teamwork

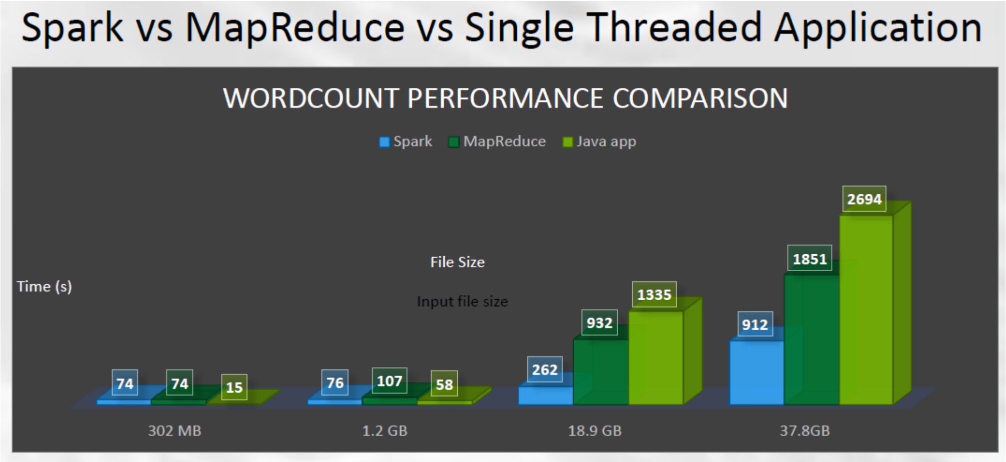
Basic "google" search on key search terms returns millions of results, many are not data related

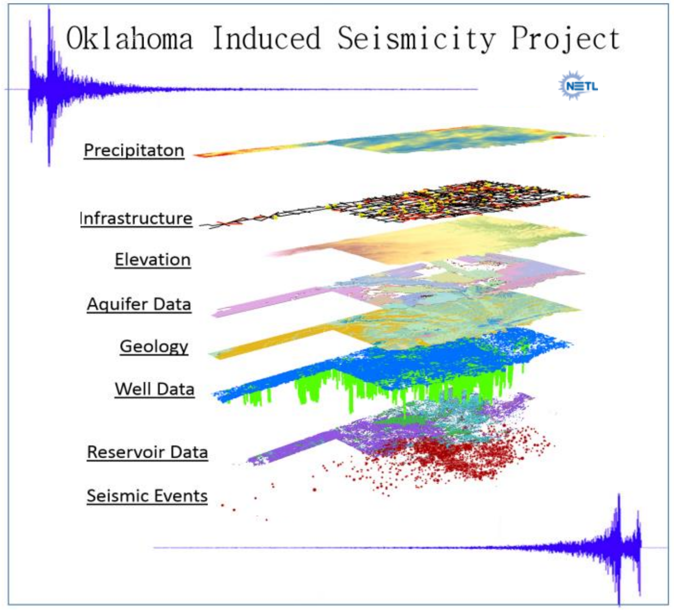


EDX Search is focused on energy data resources

Big data tools for small & big data challenges…

- Big Data capabilities can address common R&D challenges associated with data:
  - **Gathering/search**
  - **Integration**
  - **Management**
  - **Analysis/Use**
- Combined with appropriate hardware, has the potential to offer EDX users functionality to support management and R&D analytical needs





What is “big data”?

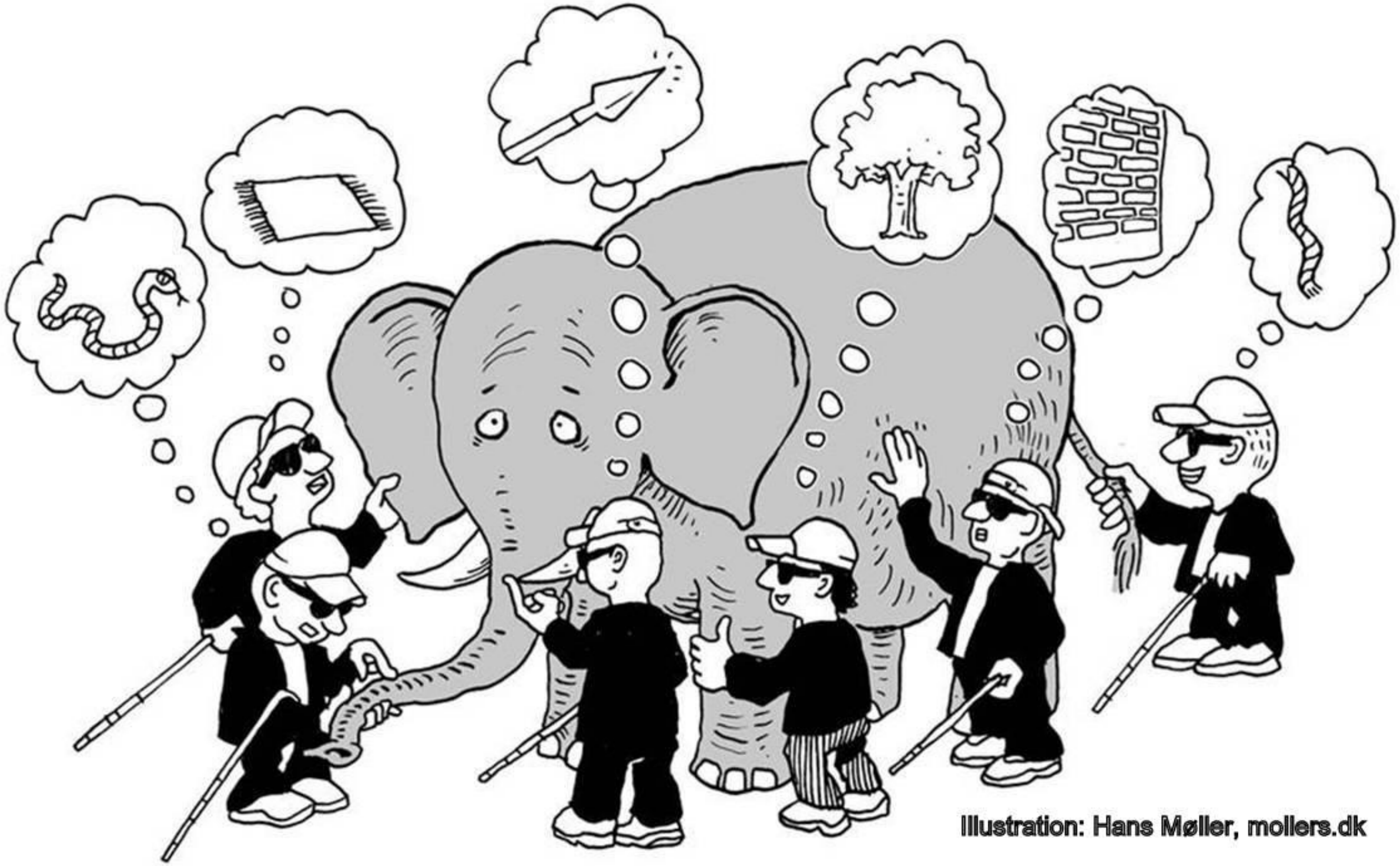


Illustration: Hans Møller, mollers.dk

What is big data computing?

- **Combination of hardware & software technologies that make it possible to realize value from "Big Datasets"**
- **HPC vs BDC**
  - Traditional **HPC** systems are focused on performing calculations at fast speeds
  - **BDC** is focused on computing to sift through huge amounts of big datasets
  - **HPC** systems usually cost $1000's of k
  - **BDC** can operate on range of hardware, including inexpensive ($10's of k) clusters optimized for distributed, in-memory, iterative processing for analytics, query, and data mining
- **Both HPC and BDC can harness cloud server farms or add additional physical nodes**

Why use big data computing?

- **Discovery, Data Mining, and Cataloging**
  - Sift through massive collections of unstructured data from multiple sources
    - Web crawling,
    - document parsing,
    - geospatial file/service processing (# features, envelope, projection, metadata)
  - Correlate relevant data using natural language processing and machine learning
    - Think “Amazon.com” recommendations for data instead of products
- **Spatial, Temporal, Image Data Processing**
  - Harness cluster computing to distribute complex computations
    - Quadtrees, nested grids
    - Nearest Neighbors
    - CT Scans

What is hadoop?

- Apache Hadoop is a software framework for storing Big Data and running applications on clusters of commodity hardware.
- Hadoop contains libraries enabling users to perform data analysis (SQL-like queries) or develop custom applications (Scala, Java, Python-based distributed jobs).
- Hadoop enables you to store, manage, and work with your Big Data.

Hadoop was not built for speed…

Hadoop was specially built to tackle Big Data problems

What Hadoop isn't…

- **We don't install applications on Hadoop**
  - (i.e., we don't install ArcGIS Server on Hadoop, but we can use Hadoop to store and work on GIS data)
- **Hadoop likes large files, not lots of small files**
  - network and disk access overhead will slowdown runtime performance
- **Hadoop isn't necessarily fast just because it's a cluster.. but…**
  - tools such as Spark, HBase, and Solr offer significant performance boost vs 'standard' Hadoop libraries

MapReduce: Single Pass Processing

- **Custom Hadoop applications (written in Java or Python)**
- **Single-pass execution – not iterative!**
  - Load data, process single iteration, export result
- **Typically consists of one 'Map' phase and one 'Reduce' phase**
- **Applications are highly specialized**
  - Architecture designed for one-pass computations,
  - Cases requiring multi-pass algorithms require stringing multiple one-pass applications together.
  - I/O not stored in memory between jobs
- **Suitable for batch operations such as image conversion**
- **Provides distributed computing option for developers but imposes constraints for algorithm design**

How Does MapReduce Work: Thought Example

Problem:

Four friends are playing cards. The cards spill on the floor.

Pickup and organize (by suit) the spilled deck of cards

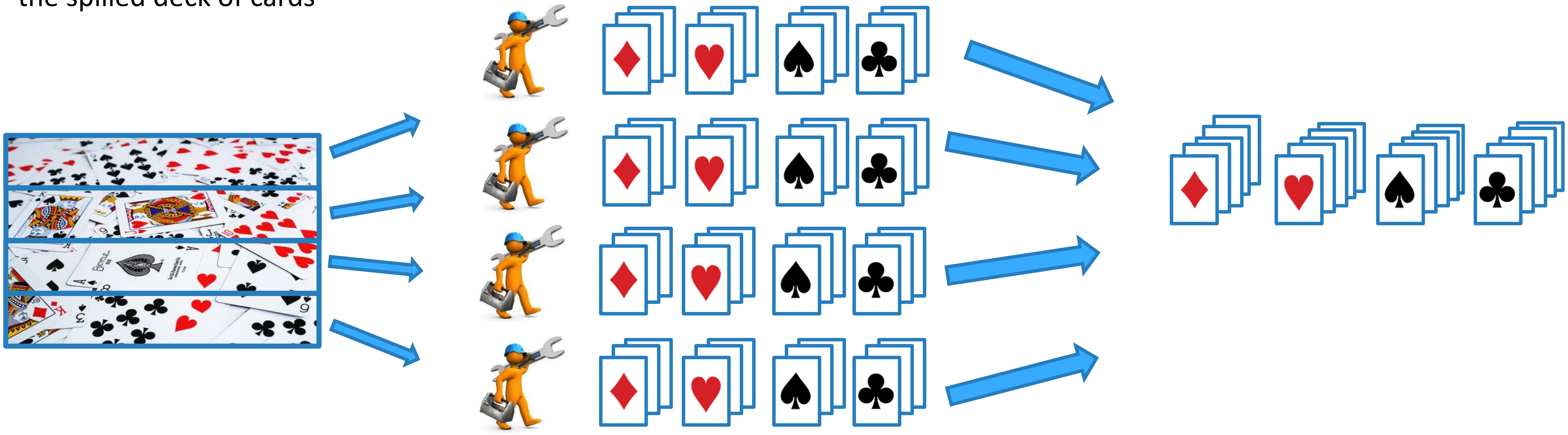
Solution:

Have each friend grab some of the cards and organize their cards by suit.

This is analogous to **'Mapping'** in Hadoop

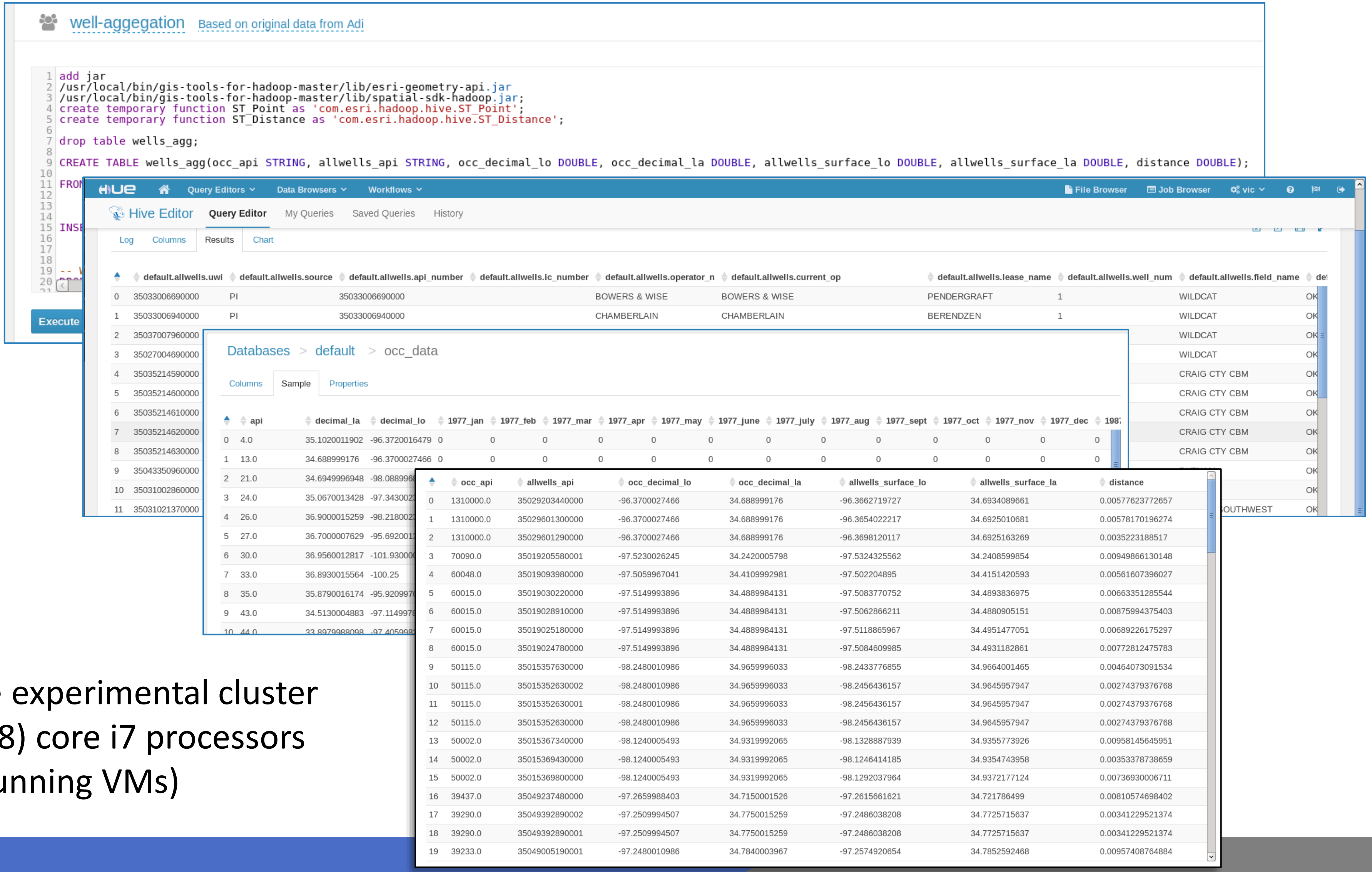
Combine each friend's stacks of cards (organized by suit) and combine like suits together.

This is analogous to **'Reduce'** in Hadoop



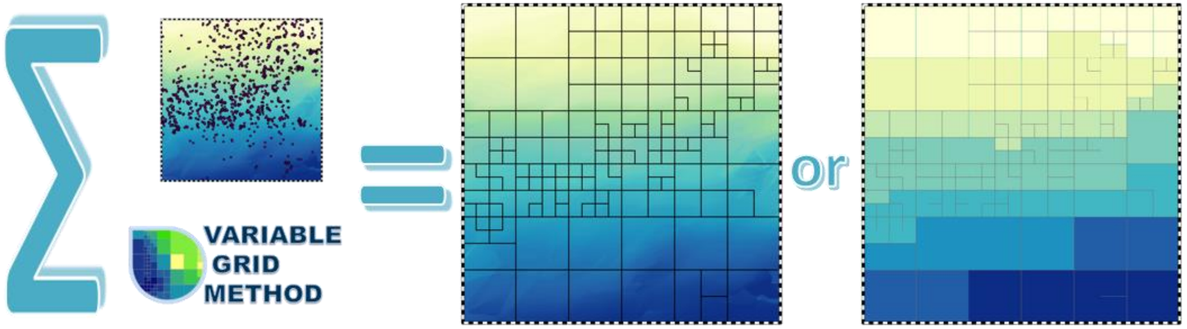
Hive Example: Well API Aggregation

- SQL-like queries on Hadoop
- Problem:
  - Well data in occ_data have bad API data
- Solution:
  - Perform spatial binning to identify nearest neighbors from valid data set using Hive
  - ***908 Million*** distance comparisons
    - 9360 OCC wells vs 97000 AllWells
- ~20 minutes using three (3) node experimental cluster consisting of desktop PCs (eight (8) core i7 processors with sixteen (16) GB RAM each running VMs)



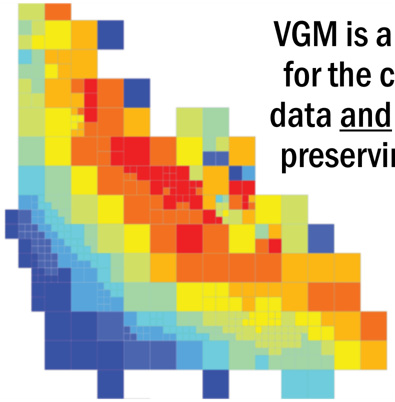
Example of big data computing for geospatial analysis:
Variable Grid Method: Capturing Uncertainty in the Analysis

VARIABLE GRID METHOD



Communicate data (via colors) and uncertainty (via grid cell size)

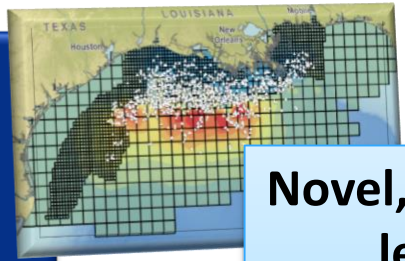
What: Variable Grid Method (VGM) is an approach designed to address issues of data uncertainty by communicating the data (colors) and uncertainties (grid cell sizes) simultaneously in a single layer.



VGM is a <u>flexible</u> method that allows for the communication of different data <u>and</u> uncertainty types, while still preserving the overall spatial trends and patterns.

Using NETL's Variable Grid Method

Communication tool to better display analytical results with their uncertainty quantification or qualification. Capable of working with various data types, formats, and uncertainty representations



Novel, flexible approach leveraging GIS capabilities to *simultaneously visualize & quantify* spatial data trends (colors) and underlying uncertainty (grid size)

VGM approach highlighted in a special issue of Transactions in GIS (July 2015)

ArcGIS, Python based **tool in beta testing** to help facilitate use of the VGM approach

Bauer, J., and Rose, K., 2015, Variable Grid Method: an Intuitive Approach for Simultaneously Quantifying and Visualizing Spatial Data and Uncertainty, Transactions in GIS-ORA-1173

Results to date – big data geoprocessing

Merging GIS and Big Data computing for advanced 3D/4D geospatial analysis

- **Offload intensive geometric operations** from desktop to a Hadoop cluster
- Is **highly scalable**
- **Self healing**
- The approach is **ideal for executing geometric operations in parallel involving many features**.

VGM use case for geoprocessing, presented at 12/2015 AGU and 6/2016 ARMA Symposium

Hadoop-Based VGM Detailed Workflow

VARIABLE GRID METHOD

Well data from ArcMap


```
"hasM": false,
"spatialReference": {"wkid":4326},
"features": [
{
"attributes": {
    "UWI__APINu": 3708520259.0,
    "OR_Base_m_": 0.0,
    "Surf_Lat": 41.484683,
    "Salinity__": 0.0,
    "WSN": 1.0,
    "Surf_Lon": -80.103193,
    "Brine_Dens": 0.0,
    "OR_Gross_T": 0.0,
    "Porosity__": 0.0,
    "NET_THICKN": 0.0,
    "Oriskany_T": 1190.549
},
"geometry": {
    "y": 5084098.520442805,
    "x": -8917047.03754837,
    "spatialReference": {
        "wkid": 4326,
        "latestWkid": 4326
    }
}
},
{
"attributes": {
    "UWI__APINu": 3703920665.0,
    "OR_Base_m_": 1077.9,
    "Surf_Lat": 41.730652,
```


Example JSON from ArcMap

VGM-Step-0

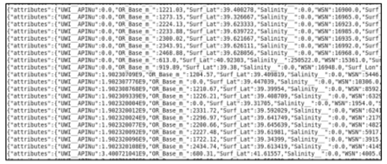
Description: Convert 'enclosed-Json' ESRI feature class into 'feature-per-row' unenclosed-Json.

Input: 'Enclosed-Json' formatted data (i.e., ORWells-wgs84.json) uploaded from ArcMap using ESRI/Hadoop toolbox tools 'Features to Json' & 'Copy to HDFS'.

Output: Processed 'Unenclosed-Json' with 'feature per row' layout suitable for Mapper.

Mapper (Setup): Create EsriFeatureClass from input file and write each feature as a row represented as unenclosed-Json.

Reducer: Aggregate Mapper output into one or more files



'Feature per row' formatted data for MapReduce

VGM-Step-1

Description: Generate multi-resolution bounding quads for input point data set (i.e., ORWells-wgs84)

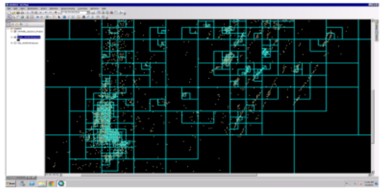
Input: vgm-step-0 output 'Unenclosed-Json' of row-per-feature representation of orwells-wgs84 data

Output: Quads of varying extents with attribution (i.e., point count, max/min/avg salinity, porosity, brine density)

Mapper (Setup): Load point features from vgm-step-0 and use to generate quadtree node extents.

Mapper: Feed mapper each row of 'unenclosed-Json' from vgm-step-0 point data and query the quadtree for all quads that contain

Reducer: Aggregate Mapper output into one or more files and store in vgm/working/output-0/.



Overlapping attributed quads (shown via ArcMap)

VGM-Step-2

Description: Generate non-overlapping topology of vgm-step-1 quads and calculate well point data per new geometries.

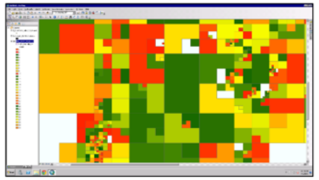
Input: Multi-resolution quads generated in vgm-step-1 AND the point data generated from vgm-step-0

Output: Non-overlapping polygons as 'unenclosed-Json' features with attribution (point count, min/max/avg porosity, etc.)

Mapper (Setup): Load vgm-step-1 output files representing attributed quads of varying resolutions to generate non-overlapping topology.

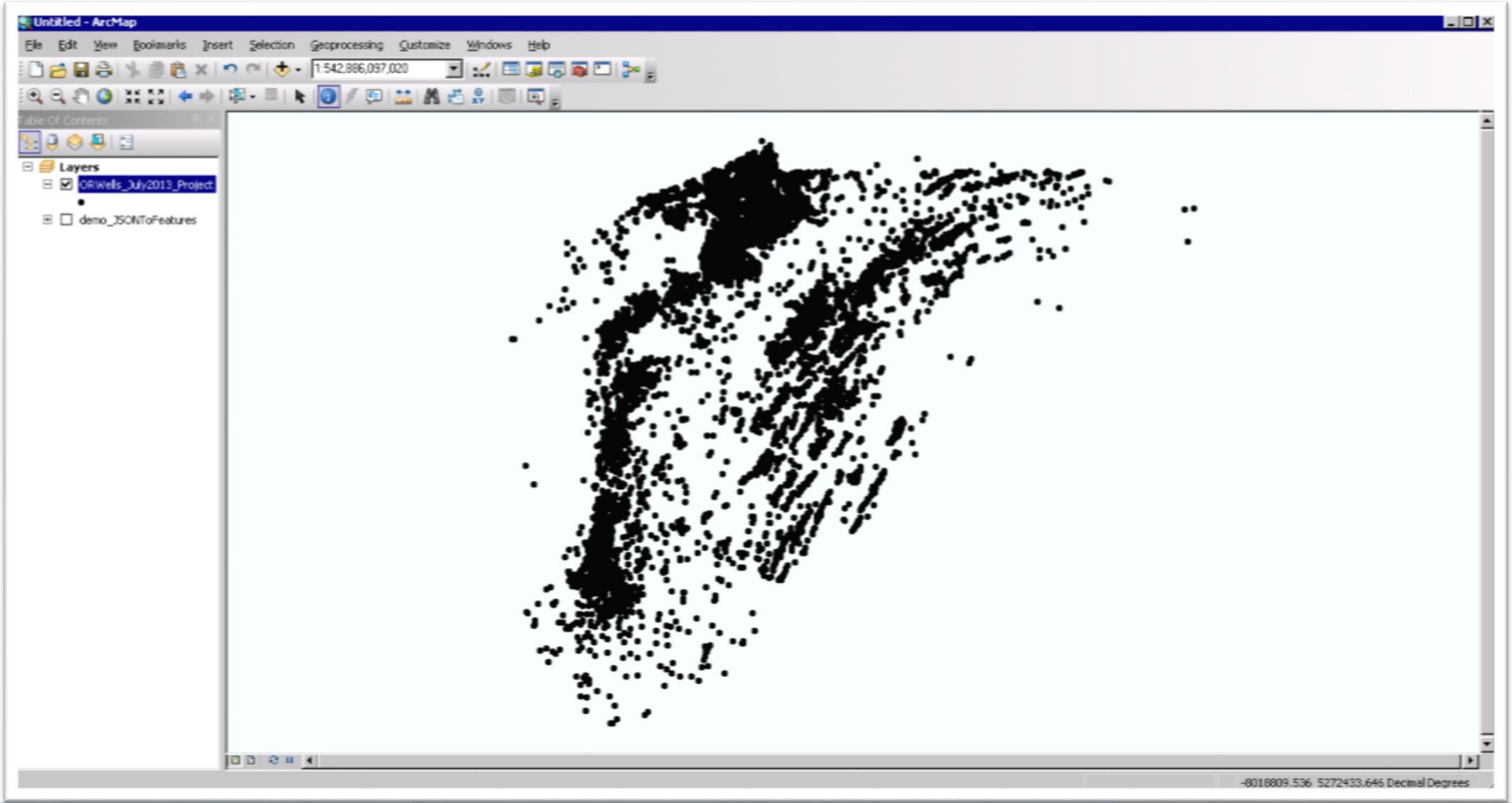
Mapper: Feed the Mapper with rows from the vgm-step-0 'unenclosed-Json' point feature data, query topology for 'point in polygon' to generate polygon's attributes, and perform geometry subtraction using ESRI Hadoop libs

Reducer: Tally the attributes for each polygon and write attributed polygon as unenclosed-Json.

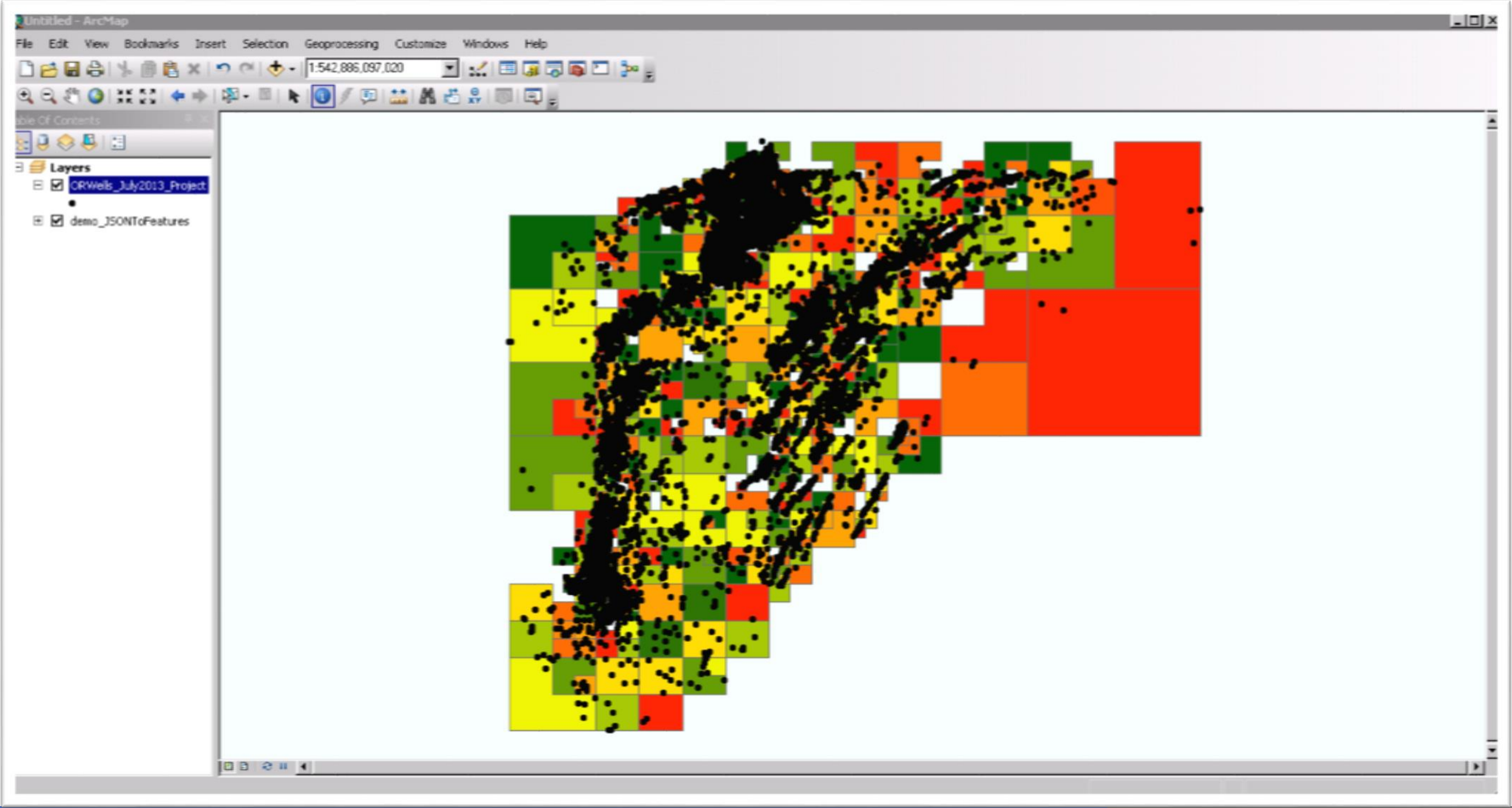


Attributed polygons for ArcMap

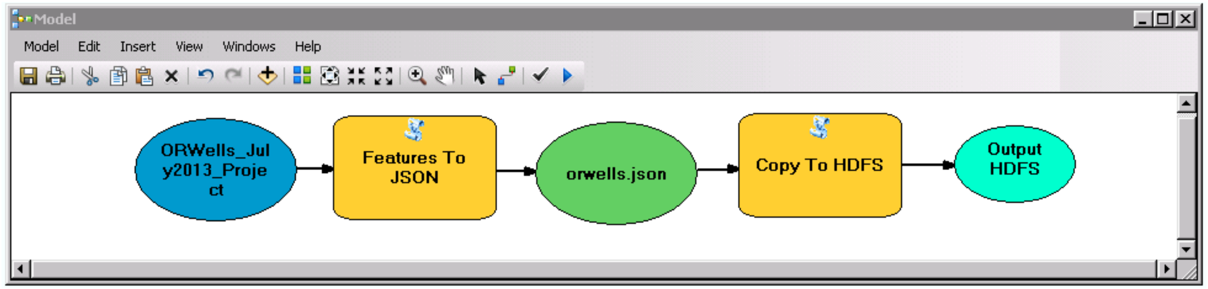
Hadoop-VGM Input



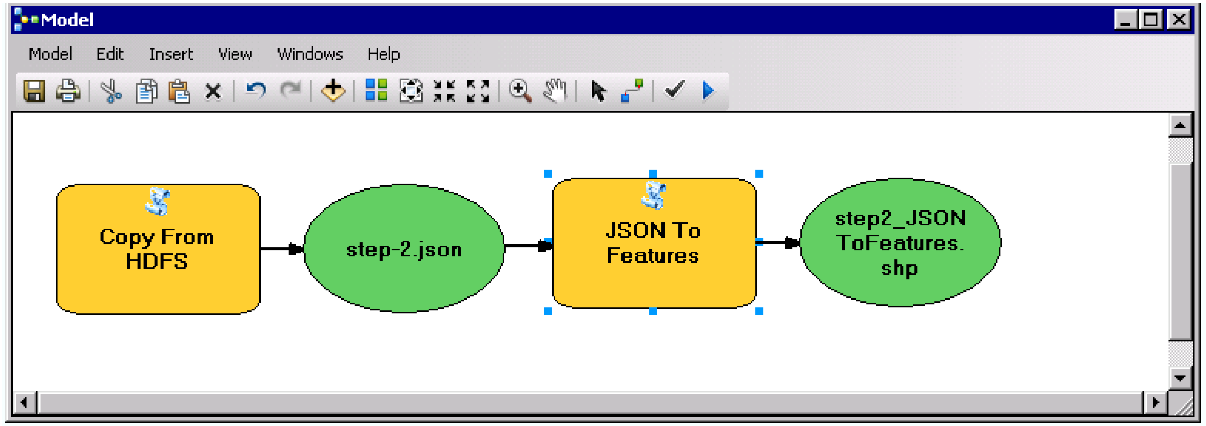
Hadoop-VGM Output



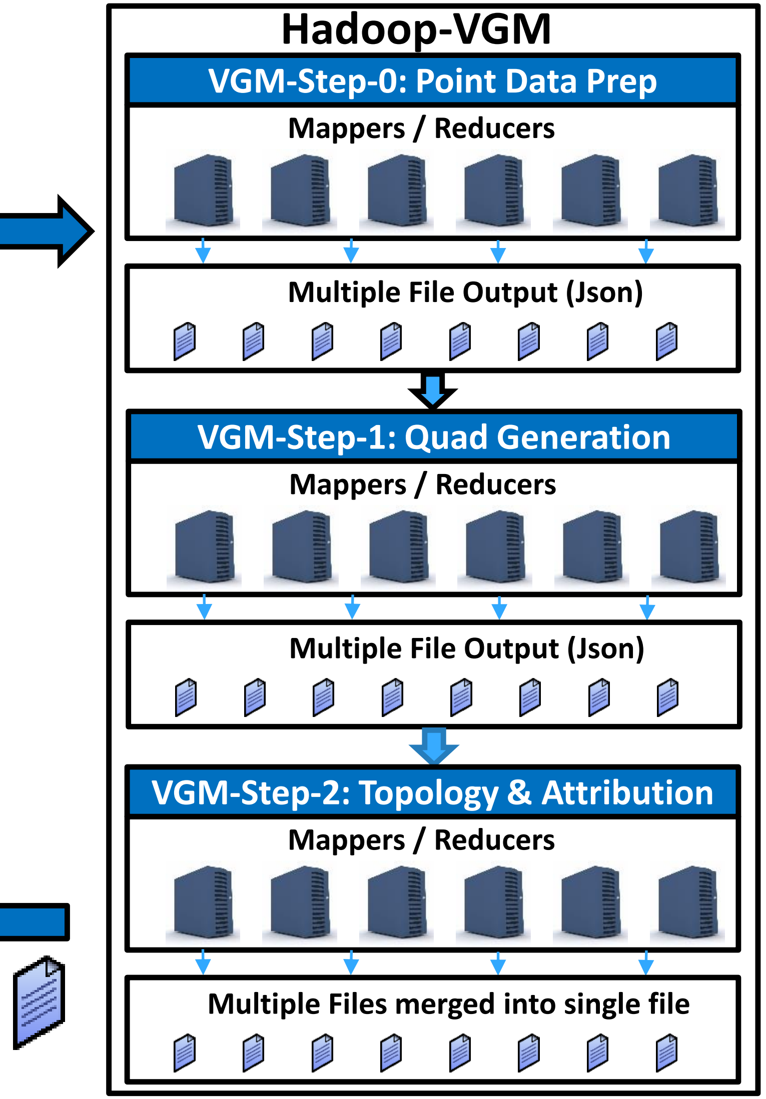
Hadoop-VGM: ArcMap to Hadoop



Model to copy ORWells point data from ArcMap to Hadoop



Model to copy results from Hadoop into ArcMap



Hadoop-VGM: Step 0: Data Prep


```
"hasM": false,
"spatialReference": {"wkid":4326},
"features": [
{
"attributes": {
    "UWI__APINu": 3708520259.0,
    "OR_Base_m_": 0.0,
    "Surf_Lat": 41.484683,
    "Salinity__": 0.0,
    "WSN": 1.0,
    "Surf_Lon": -80.103193,
    "Brine_Dens": 0.0,
    "OR_Gross_T": 0.0,
    "Porosity__": 0.0,
    "NET_THICKN": 0.0,
    "Oriskany_T": 1190.549
},
"geometry": {
    "y": 5084098.520442805,
    "x": -8917047.03754837,
    "spatialReference": {
        "wkid": 4326,
        "latestWkid": 4326
    }
}
},
{
"attributes": {
    "UWI__APINu": 3703920665.0,
    "OR_Base_m_": 1077.9,
    "Surf_Lat": 41.730652,
```


Input 'ORWells-wgs84.json' generated from ArcMap

VGM Step 0


```
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1221.03,"Surf_Lat":39.400278,"Salinity__":0.0,"WSN":16900.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1273.15,"Surf_Lat":39.326667,"Salinity__":0.0,"WSN":16965.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2224.13,"Surf_Lat":39.623333,"Salinity__":0.0,"WSN":16923.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2233.88,"Surf_Lat":39.639722,"Salinity__":0.0,"WSN":16985.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2300.02,"Surf_Lat":39.621667,"Salinity__":0.0,"WSN":16935.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2343.91,"Surf_Lat":39.626111,"Salinity__":0.0,"WSN":16992.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2468.88,"Surf_Lat":39.628056,"Salinity__":0.0,"WSN":16968.0,"Surf_
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":613.0,"Surf_Lat":40.92303,"Salinity__":250522.0,"WSN":15361.0,"Sur
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":919.89,"Surf_Lat":39.38,"Salinity__":0.0,"WSN":16948.0,"Surf_Lon":
{"attributes":{"UWI__APINu":1.90230709E9,"OR_Base_m_":1204.57,"Surf_Lat":39.409819,"Salinity__":0.0,"WSN":5446.
{"attributes":{"UWI__APINu":1.902307776E9,"OR_Base_m_":0.0,"Surf_Lat":39.447039,"Salinity__":0.0,"WSN":10306.0,
{"attributes":{"UWI__APINu":1.902308768E9,"OR_Base_m_":1210.67,"Surf_Lat":39.39954,"Salinity__":0.0,"WSN":8592.
{"attributes":{"UWI__APINu":1.902309339E9,"OR_Base_m_":1226.21,"Surf_Lat":39.408709,"Salinity__":0.0,"WSN":6329
{"attributes":{"UWI__APINu":1.902320004E9,"OR_Base_m_":0.0,"Surf_Lat":39.31705,"Salinity__":0.0,"WSN":1954.0,"S
{"attributes":{"UWI__APINu":1.902320012E9,"OR_Base_m_":2331.72,"Surf_Lat":39.592029,"Salinity__":0.0,"WSN":6241
{"attributes":{"UWI__APINu":1.902320024E9,"OR_Base_m_":2296.97,"Surf_Lat":39.641749,"Salinity__":0.0,"WSN":2179
{"attributes":{"UWI__APINu":1.902320077E9,"OR_Base_m_":2200.66,"Surf_Lat":39.645639,"Salinity__":0.0,"WSN":4827
{"attributes":{"UWI__APINu":1.902320092E9,"OR_Base_m_":2227.48,"Surf_Lat":39.61981,"Salinity__":0.0,"WSN":5917.
{"attributes":{"UWI__APINu":1.902320096E9,"OR_Base_m_":1722.12,"Surf_Lat":39.34399,"Salinity__":0.0,"WSN":3915.
{"attributes":{"UWI__APINu":1.902320108E9,"OR_Base_m_":2434.74,"Surf_Lat":39.613419,"Salinity__":0.0,"WSN":4145
{"attributes":{"UWI__APINu":3.400721041E9,"OR_Base_m_":680.31,"Surf_Lat":41.61557,"Salinity__":0.0,"WSN":4005.0
```


VGM-Step-0 output as Hadoop mapper friendly 'feature-per-row' unenclosed Json

Hadoop-VGM: Step 1: Generate Quads

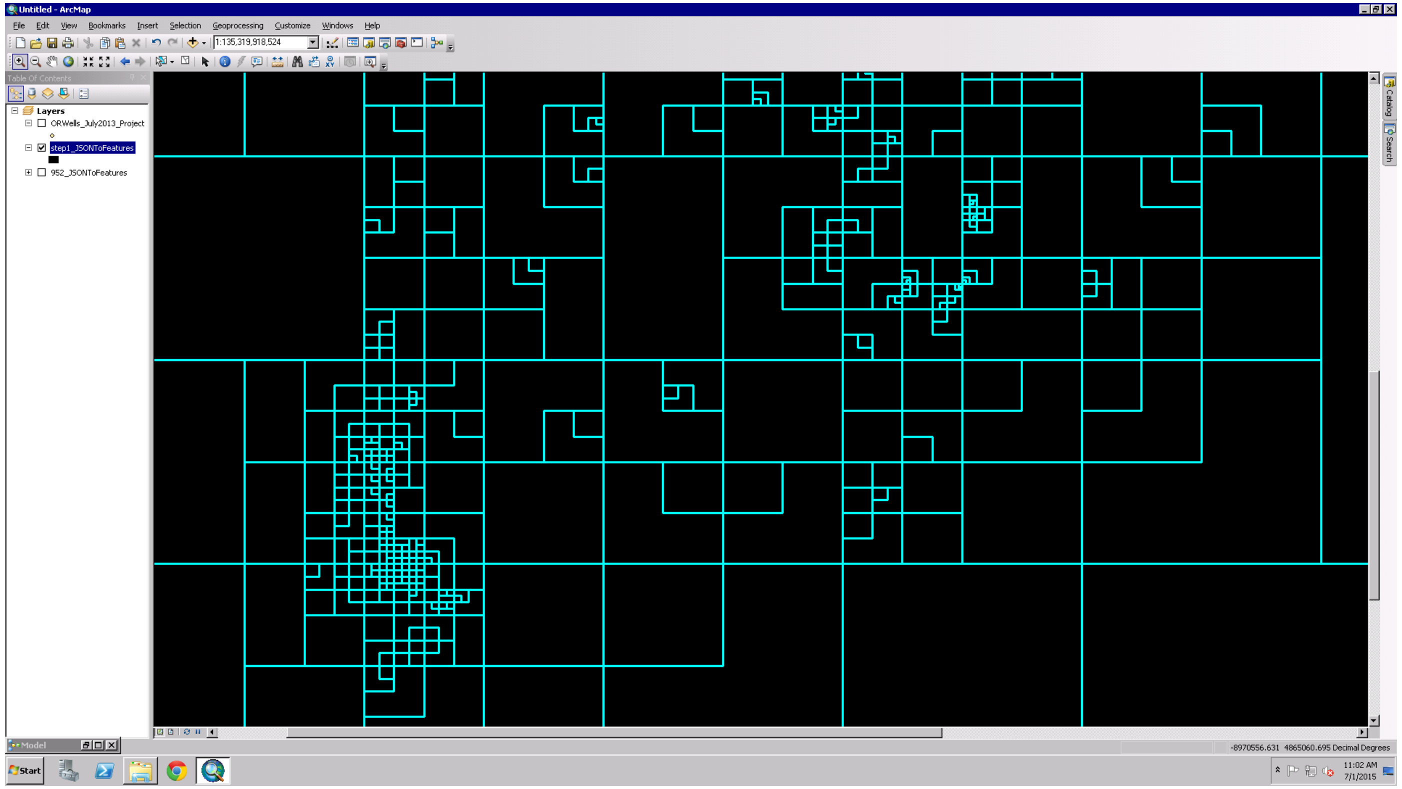
Input: VGM-Step-0 formatted point data


```
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1221.03,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1273.15,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2224.13,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2233.88,"Surf_Lat"
```


VGM-Step-1

Output: Attributed overlapping quads


```
{"attributes":{"count":10,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":23,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":20,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":13,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":14,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":15,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":10,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":16,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":23,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":10,"sumPorosity":8.3,"avgPorosity":0.83000000
{"attributes":{"count":10,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":64,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":24,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":18,"sumPorosity":3.5,"avgPorosity":0.19444444
```

(Output from this intermediate step shown in ArcMap)

Hadoop-VGM: Step 2: Generate Topology

Input: VGM-Step-0 formatted point data

{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1221.03,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":1273.15,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2224.13,"Surf_Lat"
{"attributes":{"UWI__APINu":0.0,"OR_Base_m_":2233.88,"Surf_Lat"

+

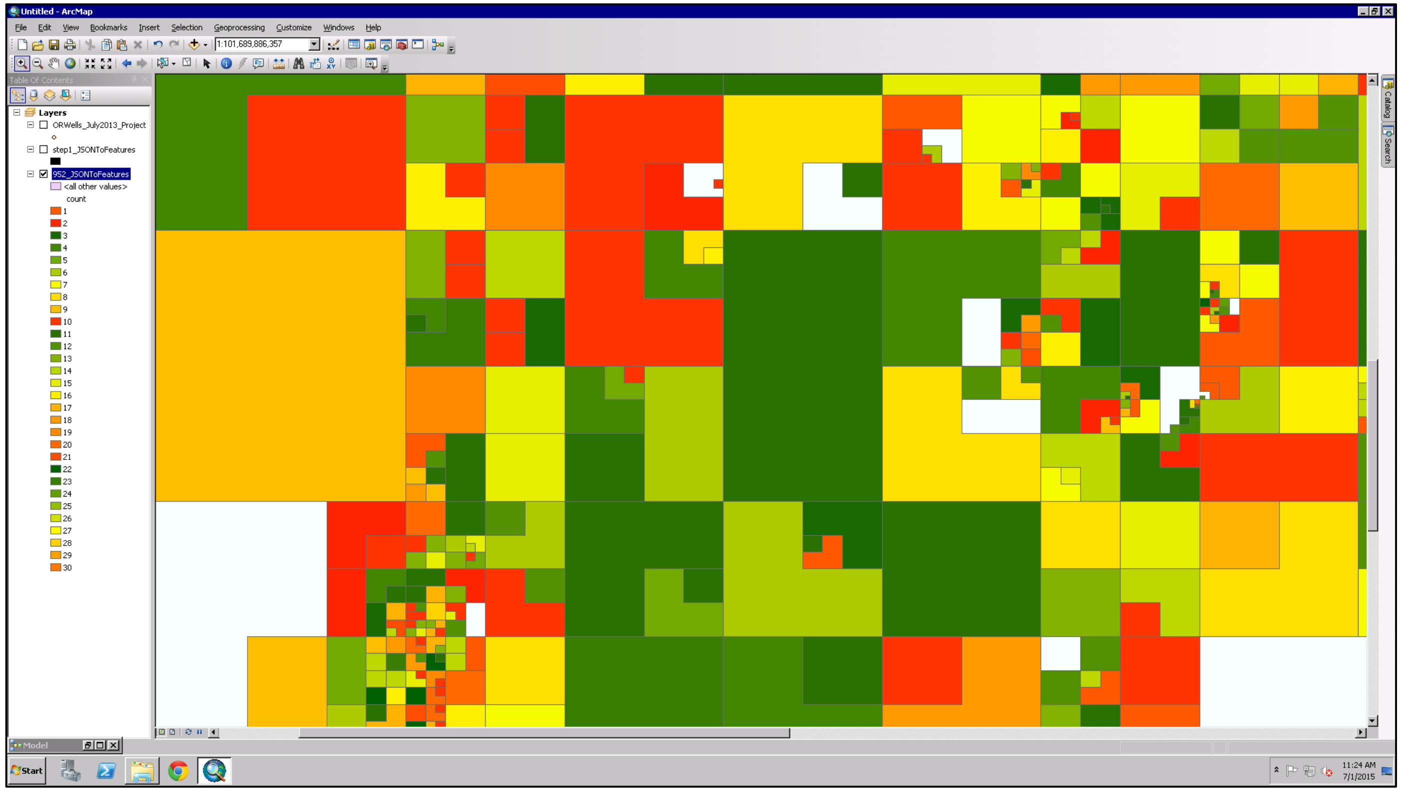
VGM-Step-1 overlapping quads

{"attributes":{"count":10,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":23,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":20,"sumPorosity":0.0,"avgPorosity":0.0,"minPo
{"attributes":{"count":13,"sumPorosity":0.0,"avgPorosity":0.0,"minPo

Output:

Updated Attribution non-overlapping polygons

{"attributes":{"count":10,"sumPorosity":33.2,"avgPorosity":3.320000
{"attributes":{"count":5,"sumPorosity":8.3,"avgPorosity":1.66000000
{"attributes":{"count":13,"sumPorosity":3.7,"avgPorosity":0.2846153
{"attributes":{"count":2,"sumPorosity":9.0,"avgPorosity":4.5,"minPo
{"attributes":{"count":12,"sumPorosity":18.0,"avgPorosity":1.5,"min
{"attributes":{"count":1,"sumPorosity":8.3,"avgPorosity":8.3,"minPo
{"attributes":{"count":20,"sumPorosity":60.7,"avgPorosity":3.035,"m



(VGM-Step-2 output in ArcMap w/ symbology based on point count)

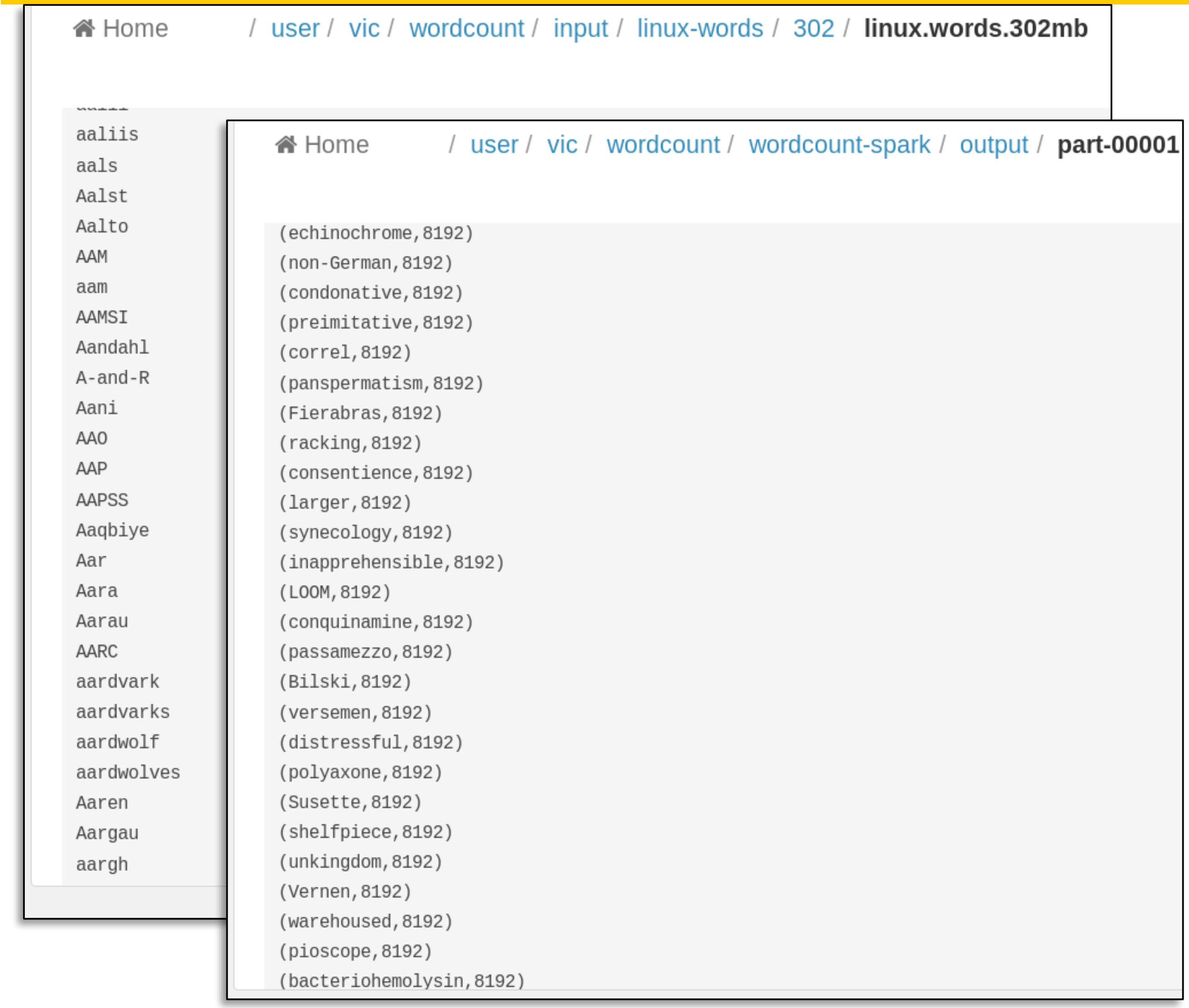
Hadoop-VGM Performance Test

- Once a working system was achieved, the next step was to scale up the amount of input data to identify opportunities for performance enhancements within the implementation.
- **Benchmarking VGM-Hadoop steps with 1 million sample points:**
  – Input data size: 114 MB
  – VGM-Step-0: 54 seconds
  – VGM-Step-1: 2 minutes 28 seconds
  – VGM-Step-2: 3 hours 28 minutes
  – Output data size: 32.8 MB
- Successfully loaded output from vgm-step-2 into ArcMap (Input data failed to load)
- Running Hadoop-VGM using a 1 million point data set identified bottlenecks in the implemented approach -- namely in VGM-Step-2's topology generation implementation.

Apache Spark: The future

- **In-memory data analysis**
  - Enables datasets and intermediate steps to be kept in memory between iterations
    - (MapReduce datasets are loaded from disk, processed via single pass, and written to disk)
- **Faster than MapReduce by an order of magnitude of more**
- **Develop iterative algorithms**
  - Repeatedly perform operations (functions) until a condition is met (unlike MapReduce)
  - Better suited for graph / tree processing (iterative bi-directional traversal)
- **Available for Java, Python, and Scala**
- **Spark is blurring the lines between HPC and BDC**

Performance Comparison: WordCount

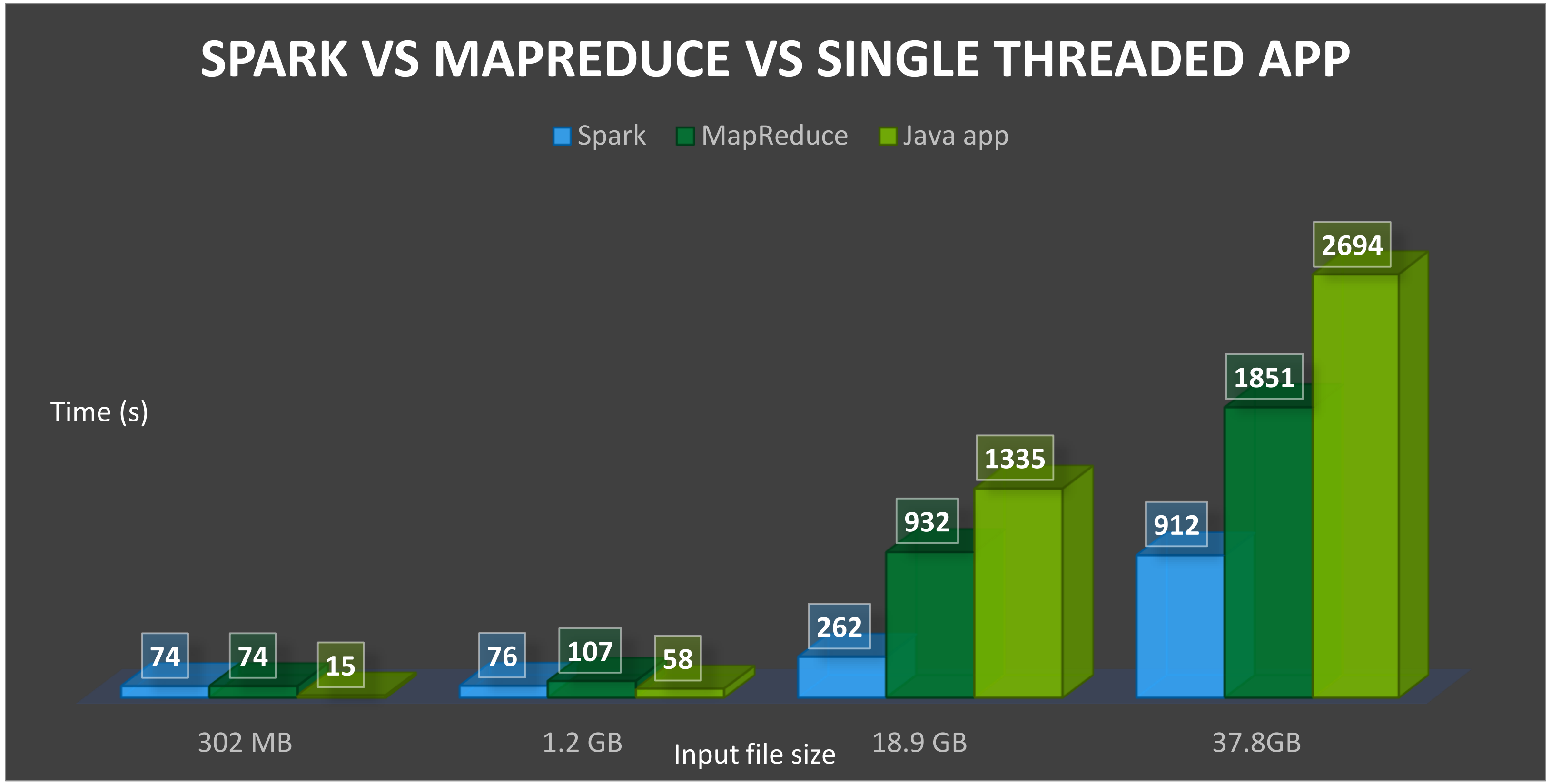


- **Word Count**
  - Count word occurrences contained within input file
  - Comparing performance for:
    - MapReduce
    - Spark
    - Stand-alone Java application
  - Input file based on copies of linux.words
  - Input file sizes:
    - 302 MB, 1.2 GB, 18.9 GB, 37.8 GB

Results to date – big data processing time test

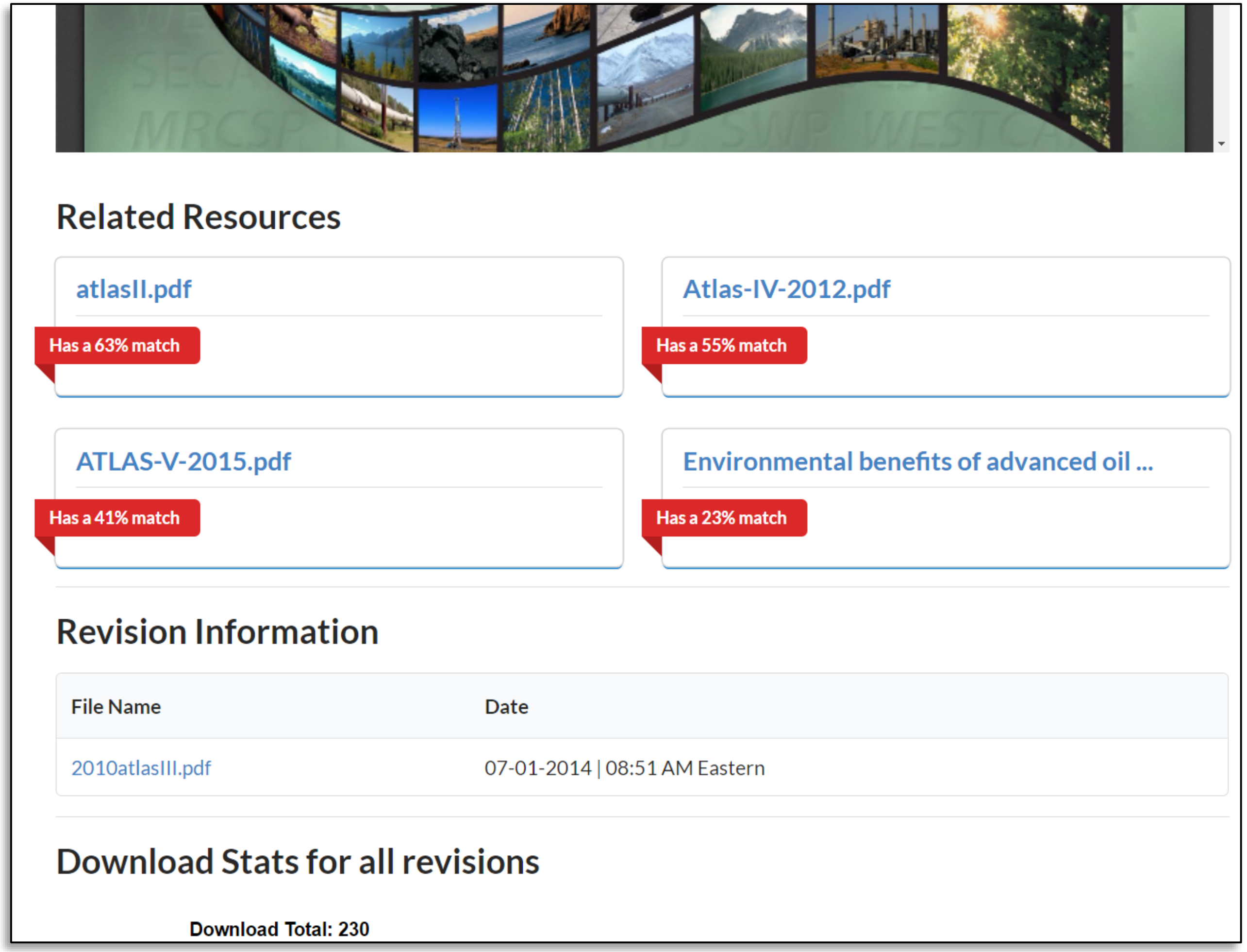
- **Compared execution times** for varying size data sets using **Hadoop** cluster-based **MapReduce** and **Spark** vs a stand alone, single threaded **Java** application (running on the Hadoop cluster's main node).

- **Spark's in-memory design outperform the single-threaded Java application for larger datasets**



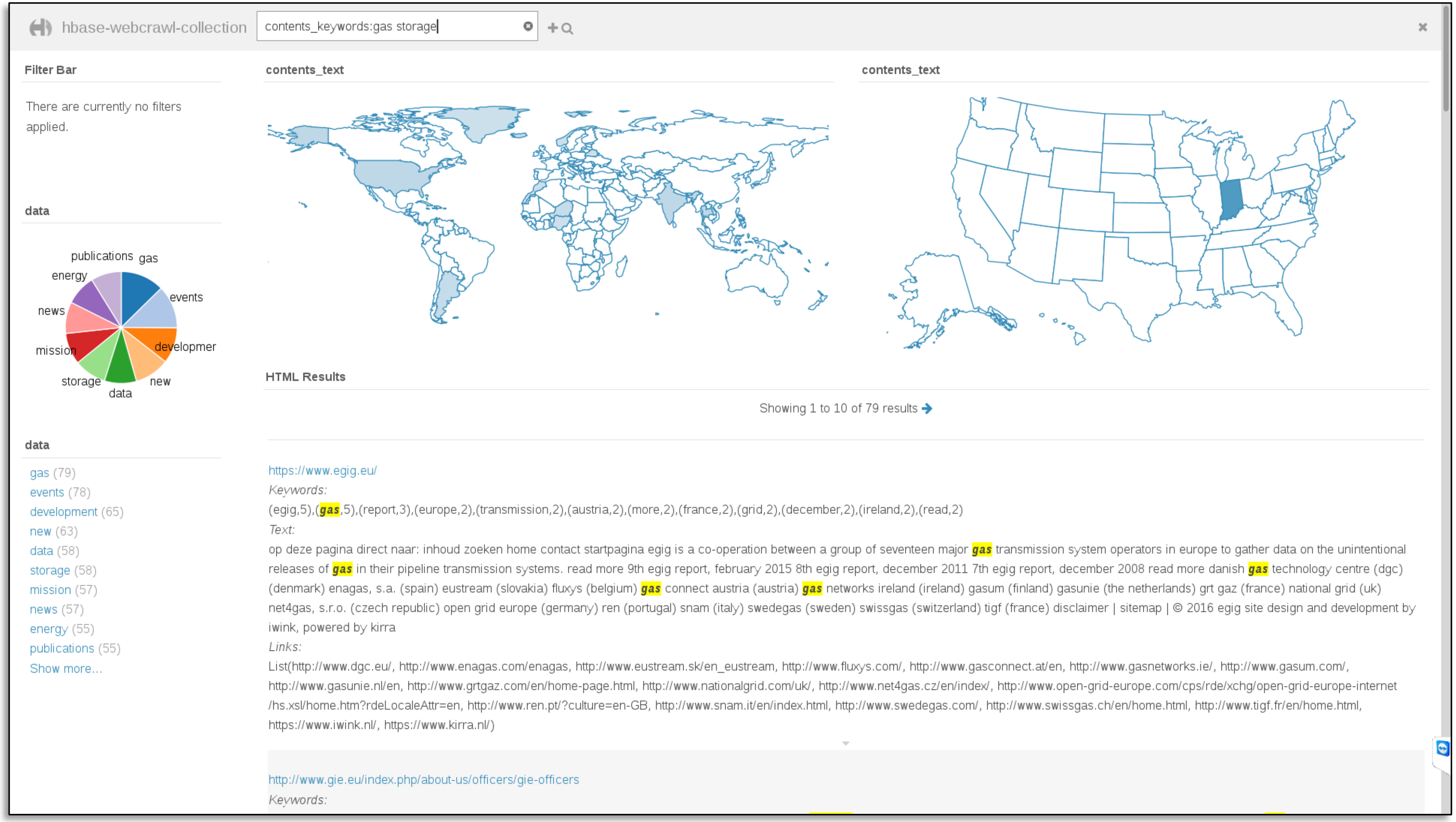
- Team succeeded in running the NN algorithm in the geoprocessing, big data cluster.
- **Time of execution went from 10 hours on desktop PC to 10 minutes**

Autoindexing: Deep Analysis Recommendation Engine



- Perform deep contextual analysis on 25k+ documents on EDX
- Machine learning, natural language processing
- Generates correlation matches of contextually similar files
- Being expanded to include spatial and webcrawl assets
- Implemented using Spark (Scala)
- Sign up @ http://edx.netl.doe.gov

Hue Dashboard for Webcrawl Application



Key Take Aways

What we've learned about BDC for geoprocessing applications:

- Capable of parallel operations; ability to scale; ideal for 'in situ' processing in the 'cloud'
- Working on integration of EDX (datasets) with geoprocessing tools & models, big data computing, and high performance computing capabilities
- Seeking to overcome to 1 million point problem (ESRI shares this problem)
- Rapidly evolving landscape – new BDC libraries and tools being released

Geoprocessing applications:

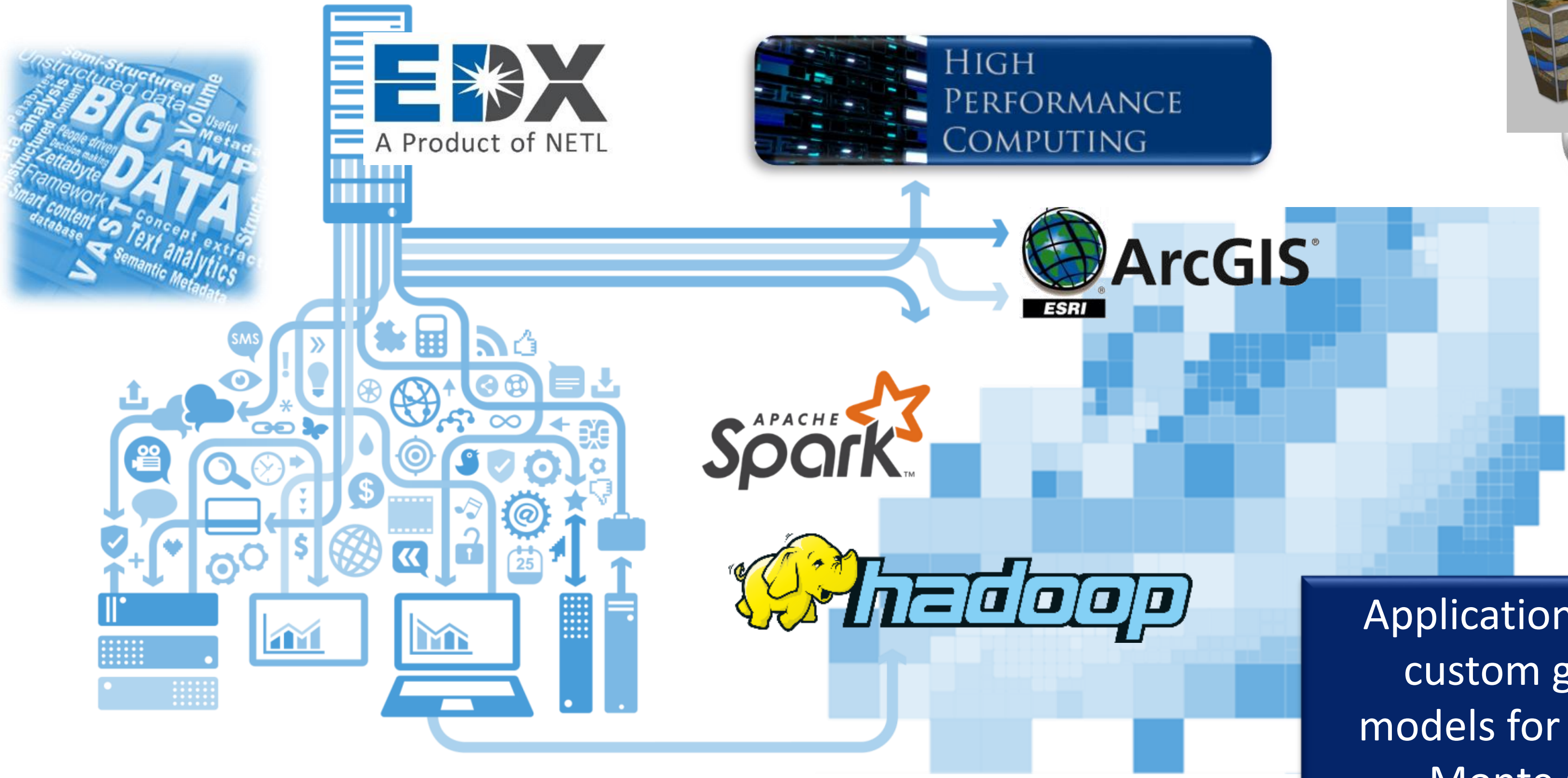
- Tested and executed improvements in geoprocessing calculation times using custom big data algorithms for i) nearest neighbor cluster analysis and ii) for uncertainty quantification/visualization approaches
- Developing custom big data search tool to improve connection of EDX users to public, authoritative datasets for energy R&D

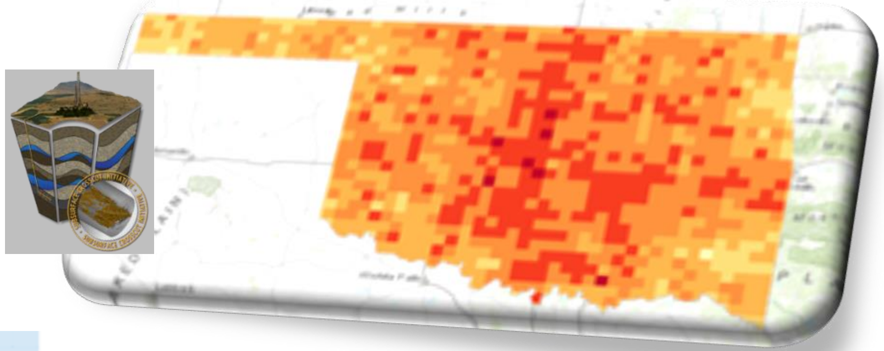
Ongoing & Next Steps

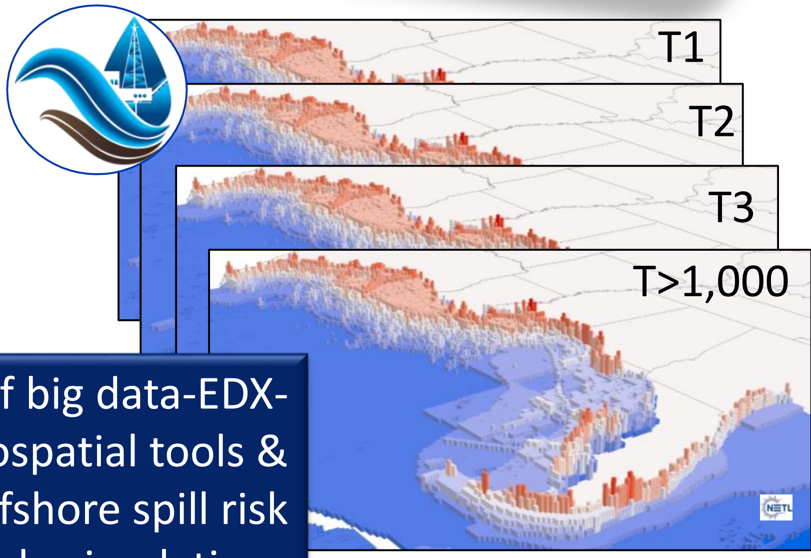
Refine & deploy through EDX big data algorithm driven search algorithm for improved data discovery

Continue developing and integrating capabilities from big data computing world, HPC arena, and GIS domain

Geoprocessing for 4D probabilistic induced seismicity risk evaluations







Application of big data-EDX-custom geospatial tools & models for offshore spill risk Monte Carlo simulations

Thank you

Vic Baker (vic.baker@matricresearch.com, vic.baker@netl.doe.gov)
Mid-Atlantic Technology, Research & Innovation Center (MATRIC),
National Energy Technology Laboratory, Morgantown, West Virginia, USA

Kelly Rose (kelly.rose@netl.doe.gov)
U.S. Dept. of Energy, National Energy Technology Laboratory, Albany, Oregon, USA

Jennifer Bauer (jennifer.bauer@netl.doe.gov)
AECOM, National Energy Technology Laboratory, Albany, Oregon, USA

Dave Rager (david.rager@netl.doe.gov)
Optimal Solutions Technologies Inc., National Energy Technology Laboratory, Morgantown, West Virginia, USA

For more information on data and tools visit:
https://edx.netl.doe.gov

EDX A Product of NETL