***WDCloud*: An End to End System for Large-Scale Watershed Delineation on Cloud**

[*] In Kee Kim, [*] Jacob Steele, [+] Anthony Castronova,
[*] Jonathan Goodall, and [*] Marty Humphrey

[*] University of Virginia
[+] Utah State University

Watershed Delineation

- **Watershed Delineation:**
  - A starting point of many hydrological analyses.
  - Defining a watershed boundary for the area of interests.

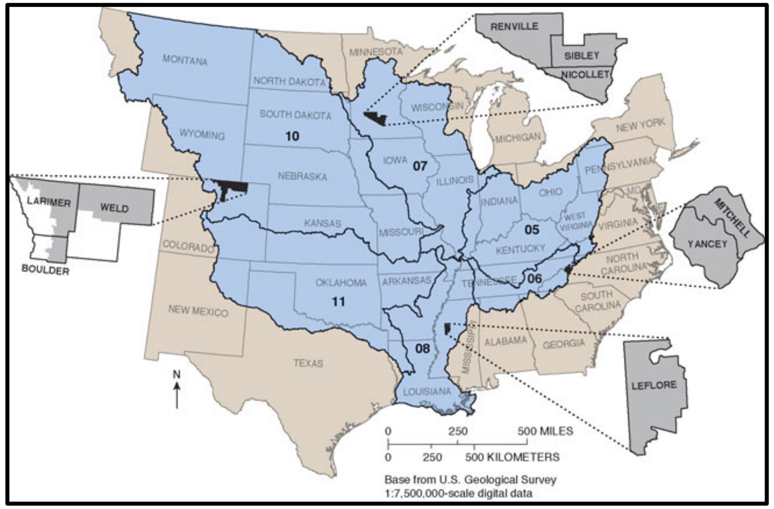
- **Why Important?**
  - Defining the scope of modeling domain.
  - Impacting further analysis and modeling steps of hydrologic research.

Approaches for Large-Scale Watershed Delineation

- **Approaches:**
  - Commercial Desktop SWs (e.g. GIS tools).
  - Online Geo-Services (e.g. USGS – StreamStats).
  - Algorithms/Mechanisms from Research Community.

- **Limitations:**
  - Steep Learning Curve.
  - Requiring Significant Amount of Preprocessing.
  - Scalability and Performance for nation-scale watersheds.
  - Uncertainty of Execution (Watershed Delineation) Time.

Research Goal



Mississippi Watershed
(Consisting of approx. 1.1 million+ catchments)

- The goals of this research is addressing
  1. The **Scalability Problem** of public dataset (NHD +)-based approach (Castronova and Goodall's approach).
  2. The **Performance Problem** of very large-scale watershed delineations (e.g. the Mississippi) using the recent advancement of computing technology (e.g. Cloud and MapReduce).
  3. The **Predictability Problem** of watershed delineation using ML (e.g. Local Linear Regression).

Our Approach

1. Automated Catchment Search Mechanism Using NHD+.

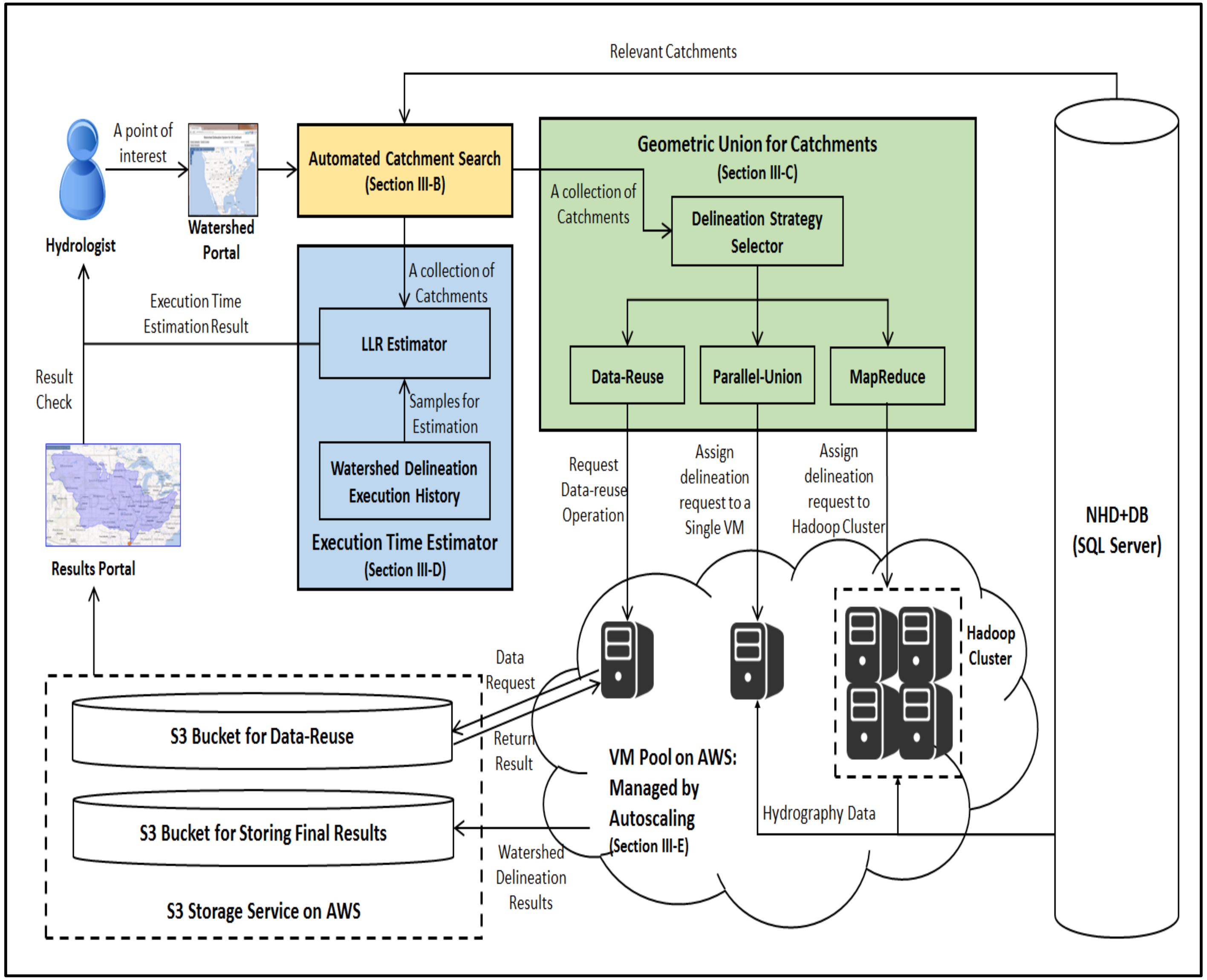
2. Performance Improvement for Computing a Large Number of Geometric Union:
    a. Data-Reuse
    b. Parallel-Union
    c. MapReduce

3. LLR (Local Linear Regression)-based Execution Time Estimation.

Our Approach

1. Automated Catchment Search Mechanism Using NHD+.
   ➔ To address the Scalability Problem.
2. Performance Improvement for Computing a Large Number of Geometric Union:
   a. Data-Reuse
   b. Parallel-Union ➔ To address the Performance Problem.
   c. MapReduce
3. LLR (Local Linear Regression)-based Execution Time Estimation.
   ➔ To address the Predictability Problem.

Design of WDCloud



WDCloud Component	Description
Web Portal for WDCloud	- Provides UI (Bing Maps) to select target watershed coordinates. - Displays the final delineation results (as well as output files (KML)).
NHD+ Dataset	- Has A single NHD+ DB (SQL Server) by integrating 21 district NHD DBs.
Automated Catchment Search Module	- Collects relevant catchments in multiple NHD regions for the target watershed.
Geometric Union Module	- Performs geometric union operation to create the final watershed.
Execution Time Estimator	- Estimate duration for the given watershed delineation via LLR.
Amazon Web Services	- Various compute resources (e.g. VMs) and storage resources (e.g Amazon S3) for WDCloud.

Automated Catchment Search Module

- Automatically search and collect all relevant catchments in multiple NHD+ regions via ***HydroSeq***, ***TerminalPath***, and ***DnHydroSeq***.

Algorithm 1 Automated Catchment Search for Multiple Regions in NHD+

```
Require: coord: coordinate for outlet of the target watershed
start_HUC_region ← get_regional_dataset (coord)
terminal_paths ← get_terminal_path_infos (start_HUC_region, coord)
catchments ← get_catchments (start_HUC_region, terminal_paths)

multi_region_hydroseqs ← get_multi_region_hydroseqs_info (start_HUC_region, terminal_paths)
if length(multi_region_hydroseqs) > 0 then
    related_HUC_regions ← find_related_HUC_regions (multi_region_hydroseqs)
    region_index ← 0
    while region_index < legnth(related_HUC_regions) do
        catchment_for_HUC_region ← get_catchments (related_HUC_regions[region_index], terminal_paths)
        catchments.append(catchment_for_HUC_region)
        region_index++
    end while
end if
```

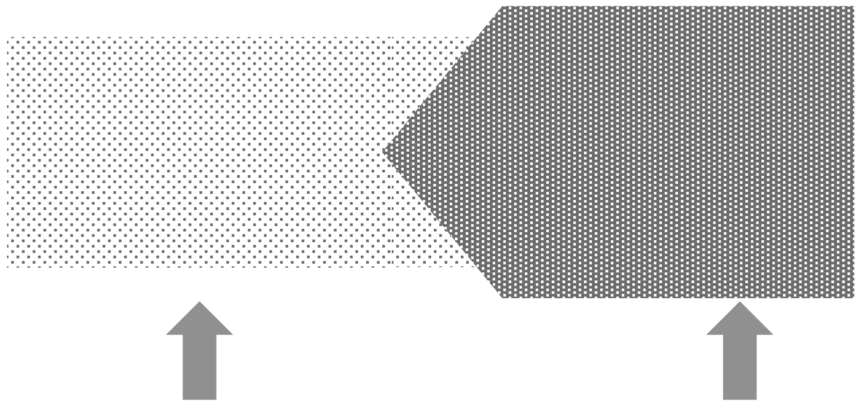
- Output: <u>*Set of Catchments that forms the target watershed.*</u>

Performance Improvement Strategies

	Strategy	Description	# of Catchments	# of VMs
Domain Specific	**Data-Reuse**	For the "monster-scale" watersheds (e.g. the Mississippi).	Multi-HUC region case. (approx. 1.1mil+)	1
System Specific	**Parallel Union**	Maximize the performance of single VM.	< 25K	1
	MapReduce	Maximize the performance of watershed delineation via Hadoop Cluster.	>= 25K	> 1

Performance Improvement – “Data-Reuse”

- **Key Idea**:
  - Pre-compute catchment unions for Monster-scale Watersheds. (not using a specific point for outlet).
  - Offline optimization to guarantee the performance of watershed delineations.

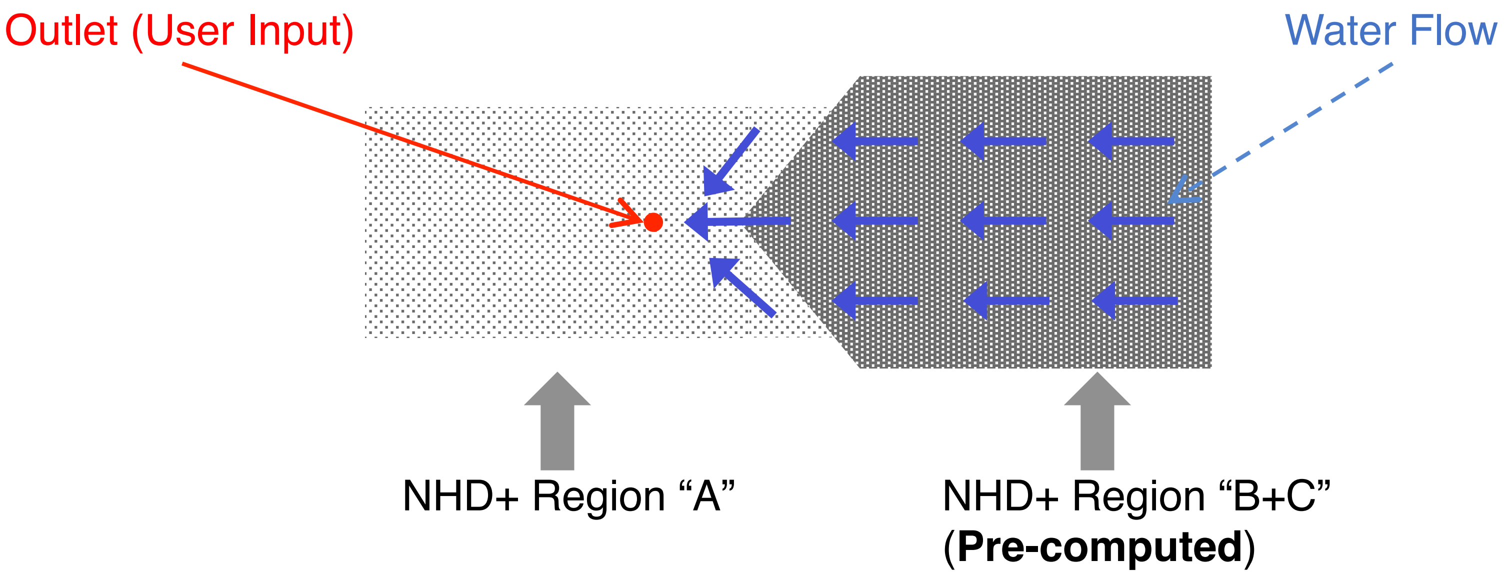


NHD+ Region “**A**”

NHD+ Region “**B+C**” (**Pre-computed**)

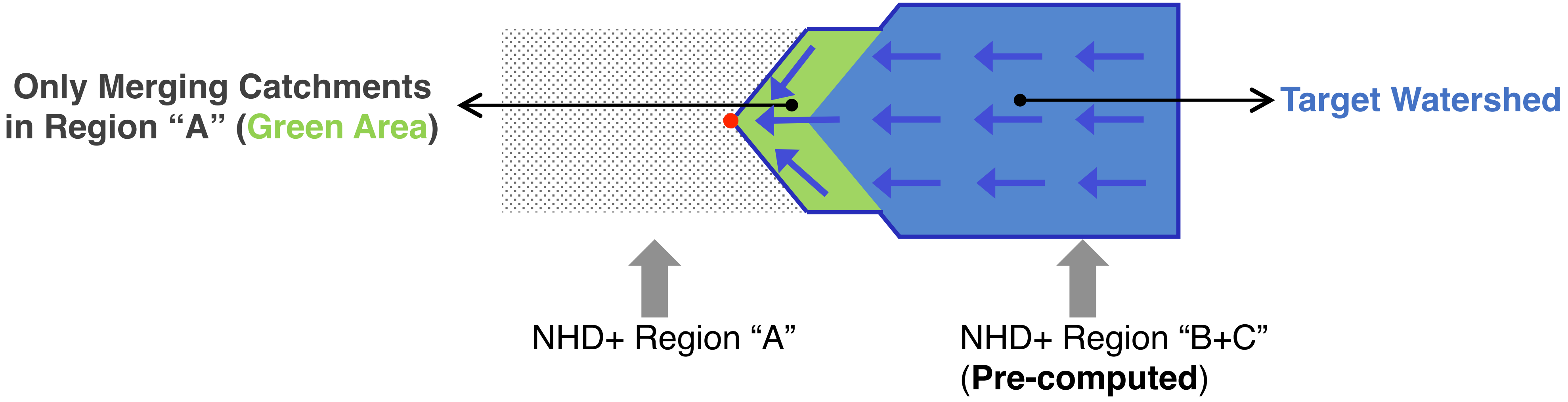
Performance Improvement – “Data-Reuse”

- **Key Idea**:
  - Pre-compute catchment unions for Monster-scale Watersheds. (not using a specific point for outlet).
  - Offline optimization to guarantee the performance of watershed delineations.



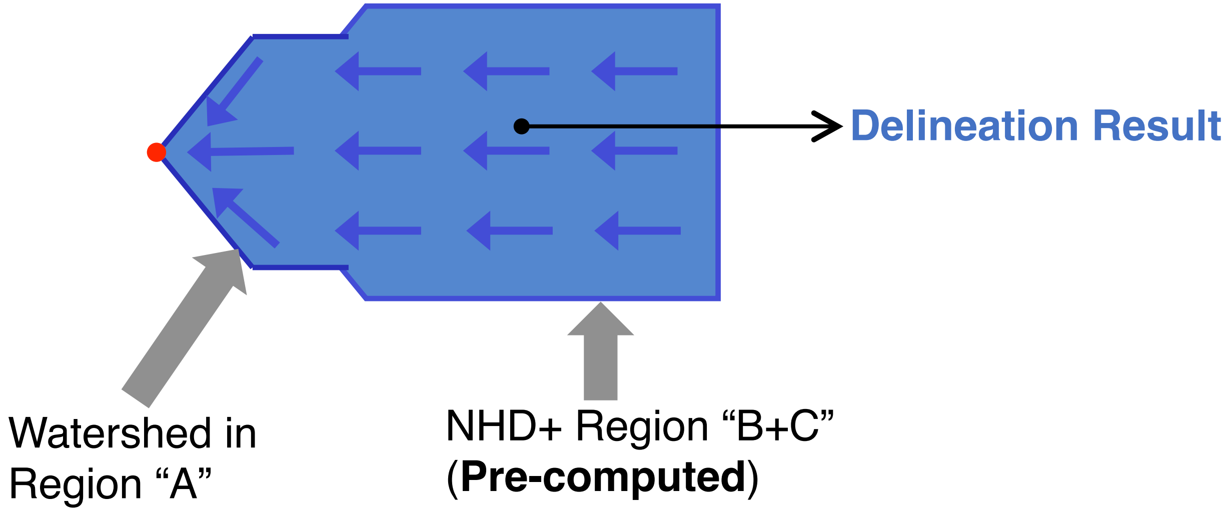
Performance Improvement – “Data-Reuse”

- **Key Idea**:
  - Pre-compute catchment unions for Monster-scale Watersheds. (not using a specific point for outlet).
  - Offline optimization to guarantee the performance of watershed delineations.



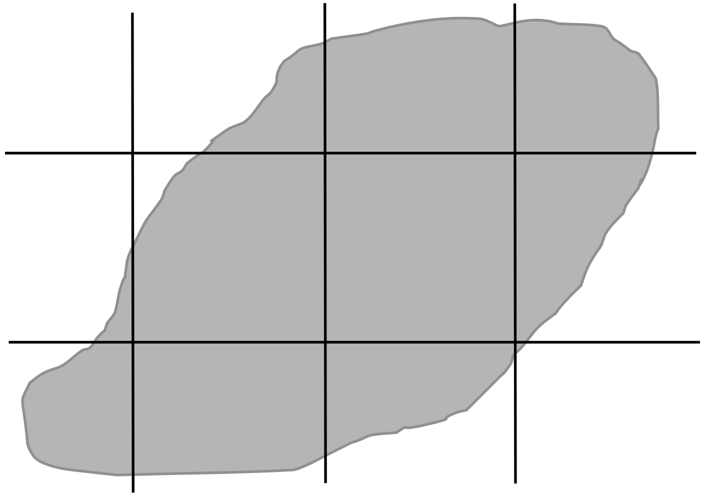
Performance Improvement – “Data-Reuse”

- **Key Idea**:
  - Pre-compute catchment unions for Monster-scale Watersheds. (not using a specific point for outlet).
  - Offline optimization to guarantee the performance of watershed delineations.

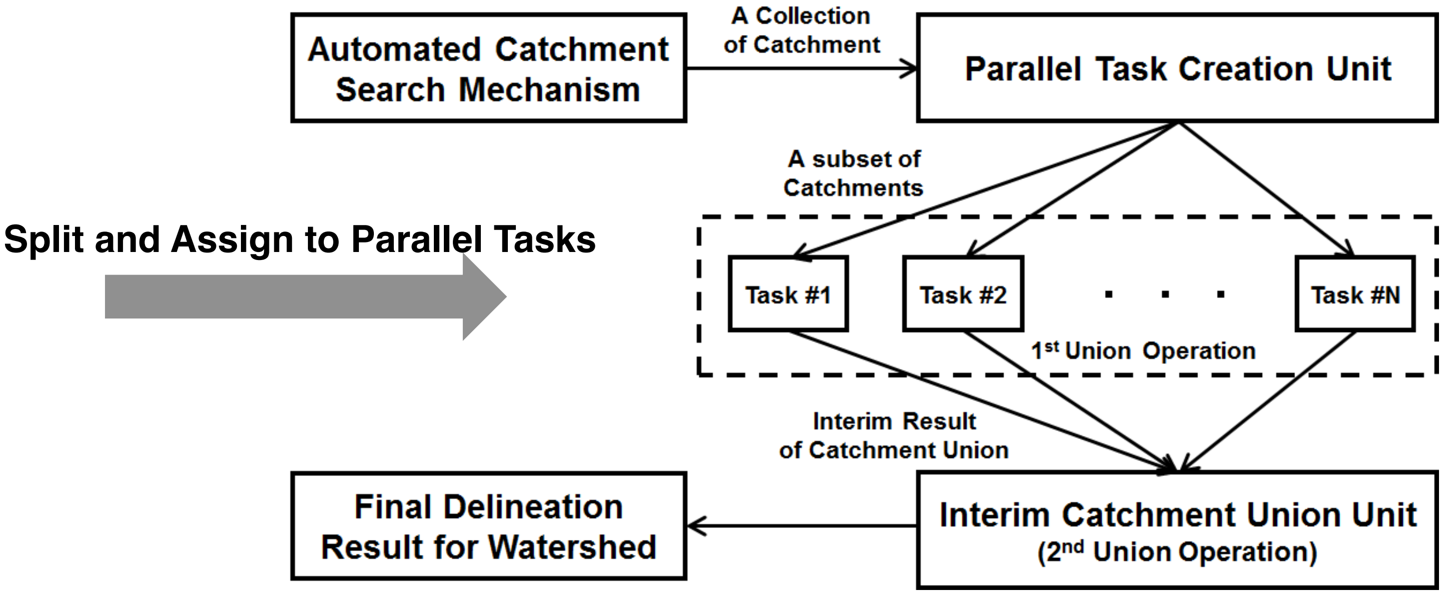


Performance Improvement – “Parallel-Union”

- **Key Idea**:
  - Used for medium-size (less than 25K catchments) watersheds.
  - Designed to maximize a multi-core (up to 32 cores) single VM instance.
  - Watershed delineation can be parallelized via “Divide-and-Conquer” or “MapReduce Style” computation.

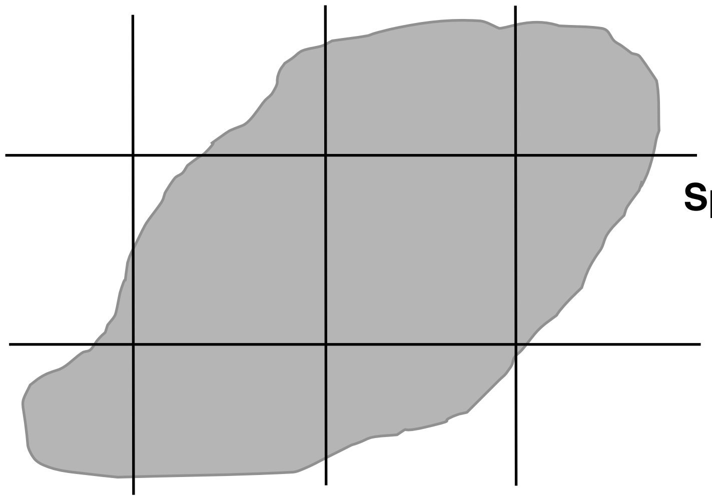


A collection of catchments for Target Watershed

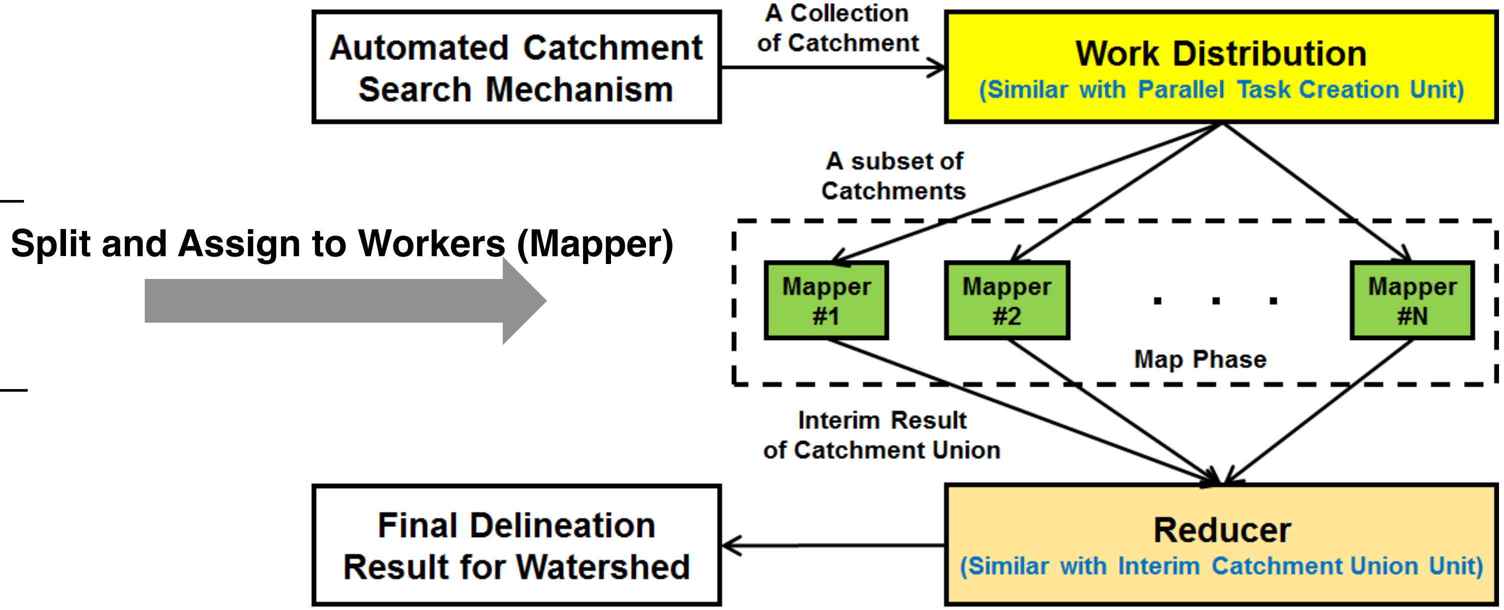


Performance Improvement – “MapReduce”

- **Key Idea**:
  - “**<u>Hadoop version</u>**” of Parallel-Union.
  - Designed to maximize the performance (minimize the watershed execution time) via utilizing multiple numbers of VM instances.
  - Used for large-size (more than 25K catchments) watersheds.



A collection of catchments for Target Watershed



Execution Time Estimation – LLR (Local Linear Regression)

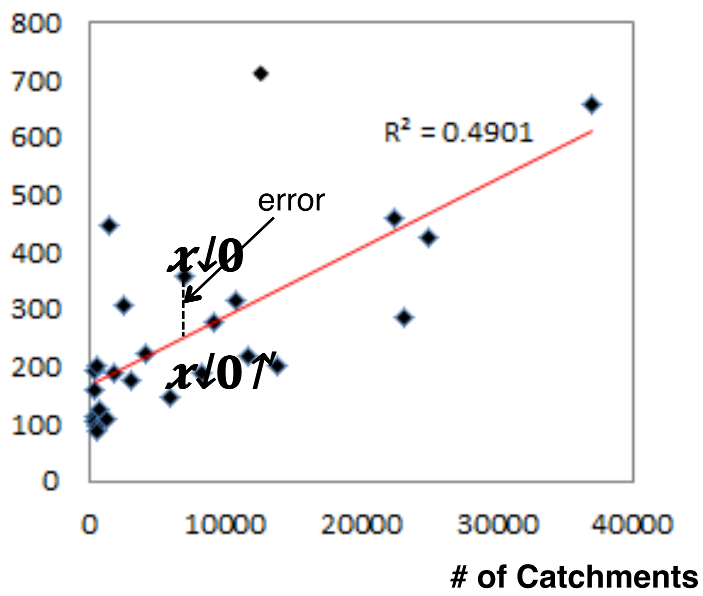
- **Initial** Hypothesis**:**
  - Execution time for watershed delineation has a ***somewhat linear relationship with IaaS/Application (Watershed Delineation Tool) specific parameters*** (e.g. VM Type, # of Catchments)
- **Watershed Delineation Tool** has several pipeline steps that each pipeline step is related to:
  - Geometric Union (Polygon Processing)
  - Non-Geometric Union
- **Data Collection and Correlation Analysis**
  - Profiled 26 execution samples on 4 different Types of VMs on AWS.

	# of Catchment	Type of VM
Non Geometric Union	**0.0973 (negligible)**	**0.7089 (strong)**
Geometric Union	**0.6129 (moderate)**	**0.3223 (weak)**

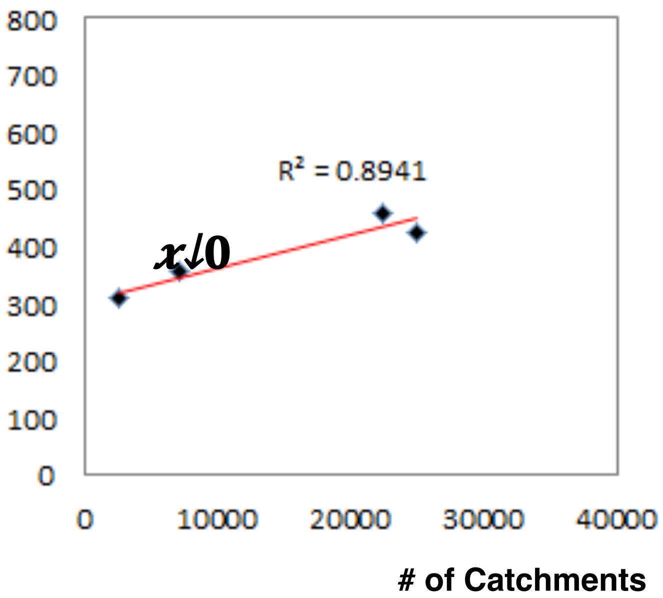
Simple Linear Model → Cannot Produce Reliable Prediction

Execution Time Estimation – LLR (Local Linear Regression)

"GLOBAL" LINEAR REGRESSION VS. "LOCAL" LINEAR REGRESSION



(a) Global Linear regression on m1.large (using all samples)



(b) Local Linear Regression on m1.large (Using three samples)

- **Procedure of Local Linear Regression**

1. Applying *k*NN to find a proper set $V(x_0)$ for prediction. → **Samples** → **2. Creating simple Regression model based on $V(x_0)$** → **Prediction Model** → **3. Making prediction for Job x_0 based on the Regression model**

- **# of Catchment**
- **Geographical Closeness**
- **Exec. Environment (VM)**

Evaluation (1) – Performance Improvement

(1) Data-Reuse
(Monster Watershed)

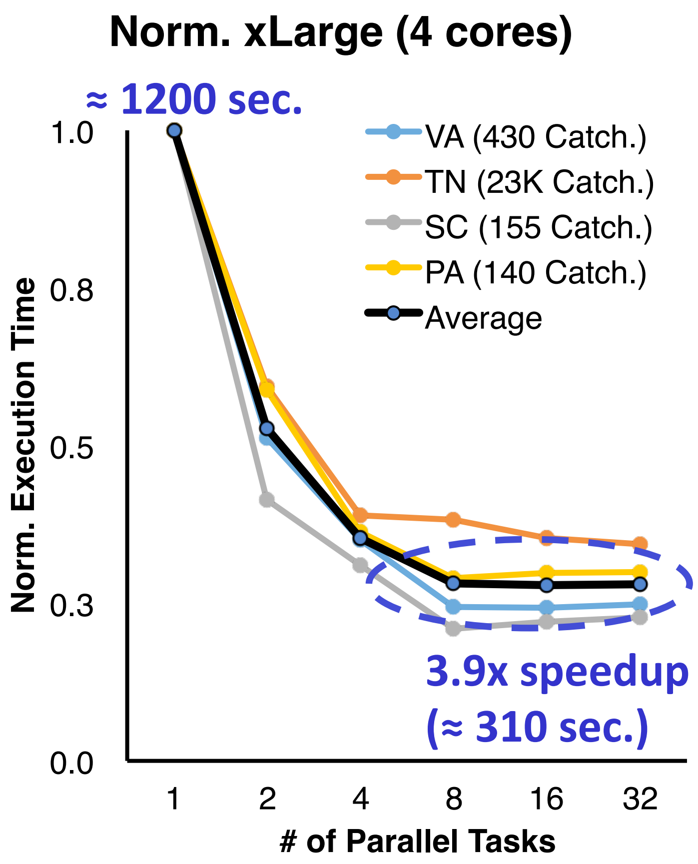
Mississippi Watershed

Comm. Desktop	Data Reuse	Speed Ups
10+ Hrs	5.5 min.	111x

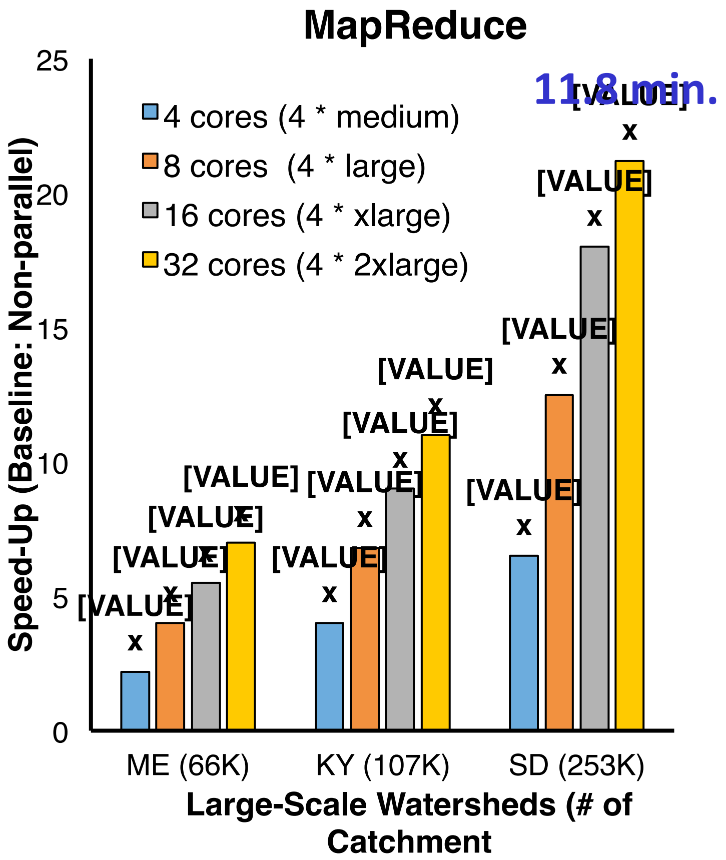
4 Core i7 with 8G RAM

M1.xlarge Instance on AWS (4 vCPUs with 7.5G Ram)

(2) Parallel-Union
(# of catch. < 25K)



(3) MapReduce
(# of catch. >= 25K)



Evaluation – Execution Time Estimation (Overall)

- Measures 420 random coordinates.
  - (20 random coordinates for watershed outlet * 21 HUC regions in NHD+)
- Metrics:

1) Prediction Accuracy

$$Prediction\ Accuracy = \begin{cases} T_{actual}/T_{predicted}, & T_{predicted} \geq T_{actual} \\ T_{predicted}/T_{actual}, & T_{actual} > T \ldots \end{cases}$$

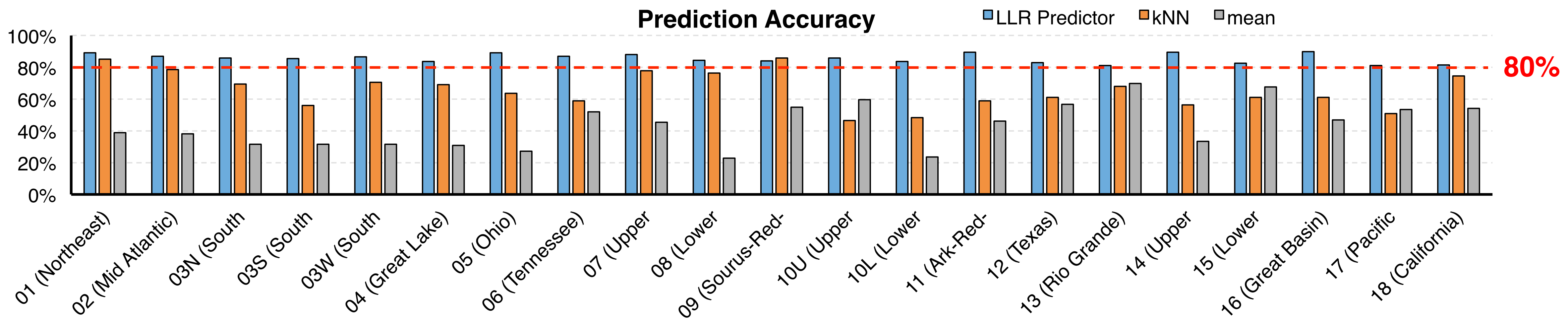
2) MAPE (Mean Absolute Percentage Error)

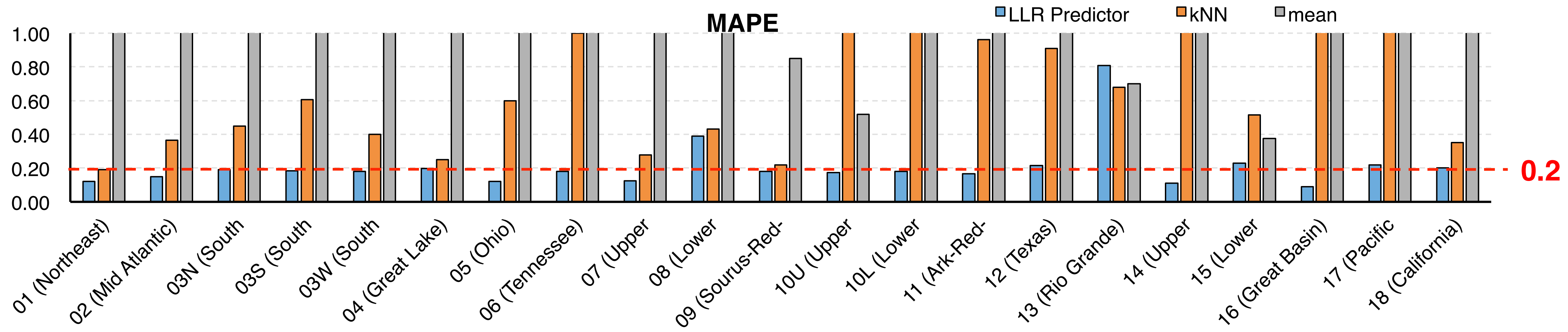
$$MAPE = \frac{1}{n} \sum_{i=1}^{n} |T_{actual,i} - T_{predicted,i} \ldots$$

Overall Results for Execution Time Estimation

	LLR Estimator	(Geo) *k*NN	Mean
Prediction Accuracy	85.6%	65.7%	42.8%
MAPE	0.19	0.93	1.97

Evaluation – Execution Time Estimation (Regional)





Conclusions

- We have designed and implemented ***WDCloud*** on top of public cloud (AWS) to solve three limitations of existing approaches:
  1) Scalability → *Automated Catchment Search Mechanism.*
  2) Performance → *Three Perf. Improvement Strategies.*
  3) Predictability → *Local Linear Regression.*

- Evaluations of ***WDCloud*** on AWS:
  - Performance Improvement
    - 4x ~ 111x speed up (Parallel Union, MapReduce, Data Reuse)
  - Prediction Accuracy
    - 85.6% of prediction accuracy and 0.19 of MAPE.

Questions?

Thank you!

Support Slides (NHD+ Regions)