EDB 2016
Emerging Database Conference 2016

SOOKMYUNG WOMEN'S UNIVERSITY

EDISON 한국과학기술정보연구원 Korea Institute of Science and Technology Information KISTI www.kisti.re.kr

Design and Implementation of a Data-Driven Simulation Service System

October 17, 2016

Ki Yong Lee, YoonJae Shin, YeonJeong Choe, SeonJeong Kim
Sookmyung Women's University

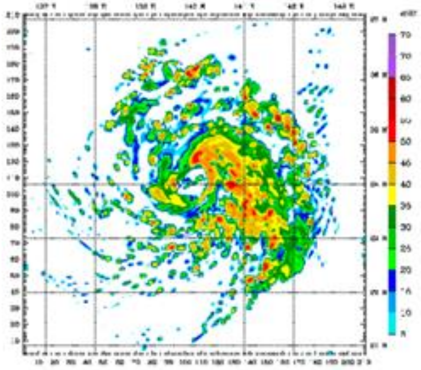
Young-Kyoon Suh, Jeong Hwan Sa, Kum Won Cho
Korea Institute of Science and Technology Information (KISTI)

Outline

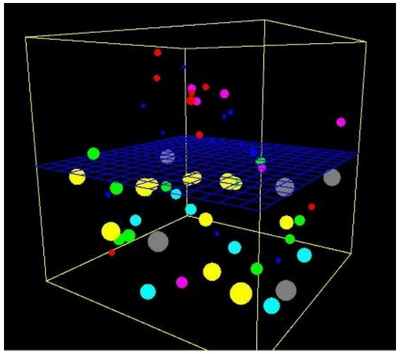
- Introduction
- Related Work
- Our Simulation Service System
- Performance Evaluation
- Conclusions

Introduction

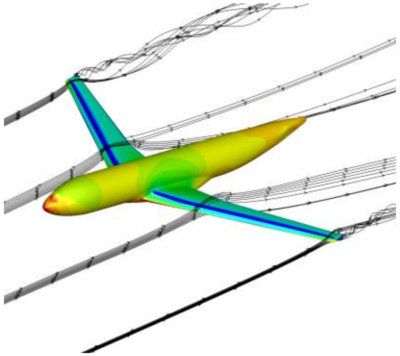
- **Computer simulations** are widely used in various fields of science and engineering
  - Computational fluid dynamics (CFD), astrophysics, particle physics, climate science, evolutionary biology, ecology, medicine, epidemiology



(a) Weather simulation



(c) Osmosis simulation

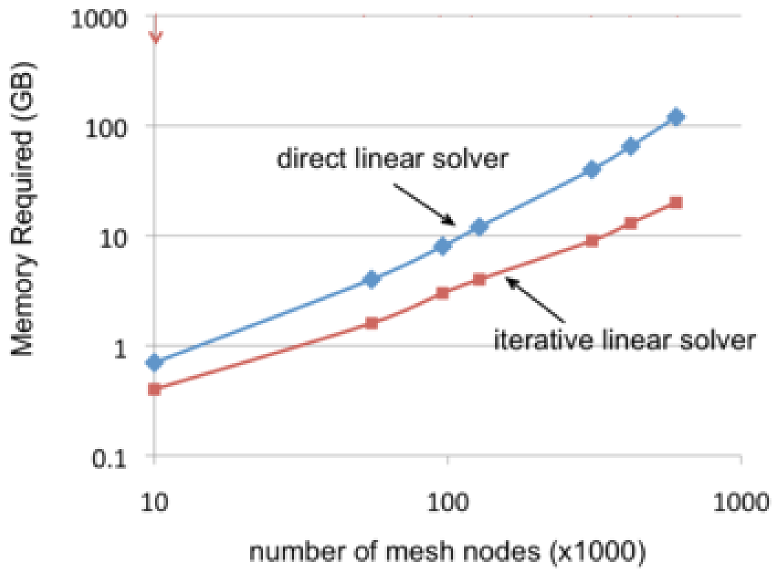


(c) Airfoil flow simulation

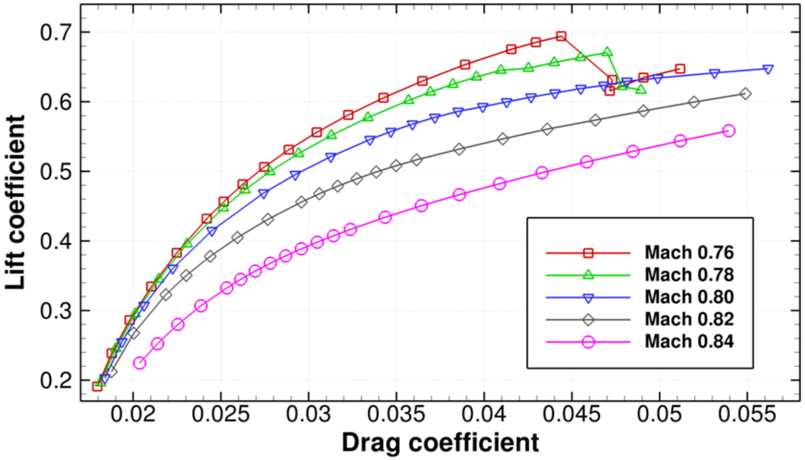
- **Simulation program**
  - **Input**: the values of the system's state variables
  - **Output**: the system's next state calculated with numerical algorithms

Increasing Cost of Simulations

- As the demand for the accuracy and quality of simulations grows, the cost of executing simulations is also ***rapidly increasing***
  - (ex) the number of equations to be solved becomes larger

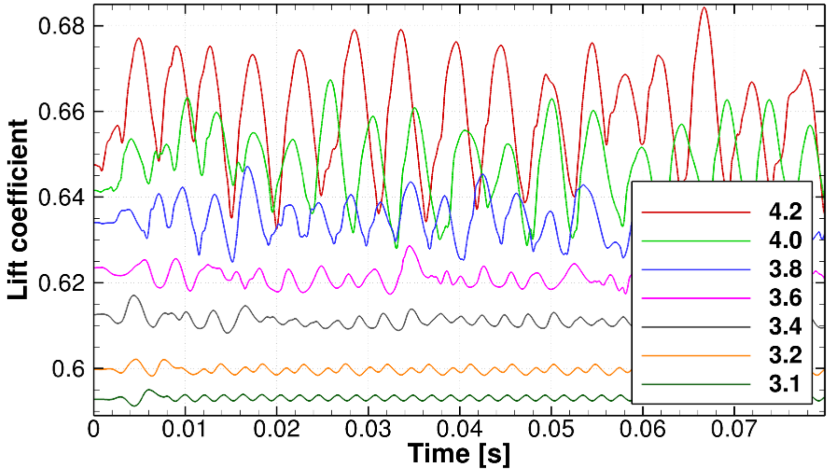


- To make matters worse, simulations are often ***repeatedly*** executed for different values of input parameters
  - In this case, the cost of simulations may be prohibitively high

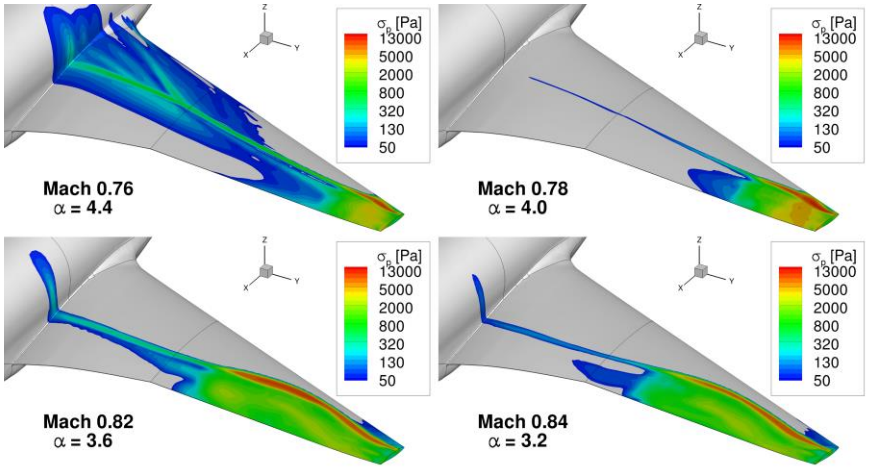


Simulations with Different Input Values

- Example: ***Airfoil simulation***
  - Users execute simulations for various values of input parameters
    - Angle of attack: 0°, 1°, 2°, ..., 15°
    - Mach number: 0.05, 0.1, 0.15, ..., 0.5
    - Reynolds number: 1×10^5, 2×10^5, 3×10^5, ..., 1×10^6
    - Other input parameters



(a) Lift coefficients for ***various values*** of angles of attack



(b) Pressure fluctuations for ***various values*** of Mach numbers

Improving Simulation Performance

- Because the cost of executing simulations is increasing, it is very important to ***reduce the cost of executing simulations***

- Approach 1: improve the performance of computer ***hardware***
  - Multi-core CPUs or Multi-core GPUs (on a single computer)
  - A computer cluster (a group of computers connected together)

- Approach 2: optimize numerical ***algorithms*** used in simulation
  - For many numerical problems, there are many alternative algorithms that vary in speed and performance
    - Equation solving, matrix decomposition, maximization/minimization, regression, clustering, etc.
  - Thus, a good algorithm is essential to reduce the execution time

Our Approach (1/2)

- Until now, the ***reuse*** of previously obtained simulation results to improve the execution of later simulations has not been much investigated yet

- Most existing simulation service systems
  - Conduct the same (perhaps long-running) simulations from scratch each time it is requested
  - Or provide only limited ability to search the previous simulation results

- If we store and utilize ***previously obtained simulation results…***
  - We can avoid redundant computations
  - We can reduce the execution time of a simulation
  - We can reduce the burden on the simulation service system

Our Approach (2/2)

- Data-driven application system (DDAS)
  - A system where execution flow is governed by data it processed
  - Obtained data can be incorporated in to the execution of the application

- In this paper, we develop a ***data-driven*** simulation service system
  - Executes requested simulations and returns the result back to the user
  - <u>Utilizes the previous simulation results to improve the execution of later simulations</u>

- The main functionality of our system
  ① Loading simulation results into the database
  ② Reusing simulation results for requested simulations
  ③ Predicting simulation results upon request

Main Functionality of Our System

① Loading simulation results

- The user can load the result of a completed simulation into the database
- ***A bulk loading feature*** is also provided

② Reusing simulation results

- If the result of a requested simulation already exists in the database, the system returns the result ***without executing the simulation again***

③ Predicting simulation results

- If the result of a requested simulation does not exist in the database, the system predicts the simulation result based on the previous data
- We employed several popular ***statistical machine learning techniques***
  - Linear regression, support vector machine, neural networks, k-nearest neighbor interpolation, decision trees, etc.

Existing Simulation Service Systems

- DataSpaces
  - Provides a shared-space abstraction for simulation data indexing and querying
- BIGNASim
  - A NoSQL database portal for nucleic acids simulation data
- SciDrive
  - A free open-source scientific data publishing platform with the simplicity of Dropbox
- DCMS (Database-Centric Molecular Simulation) system
  - Stores molecular simulation data in a relation database to query and search simulation results

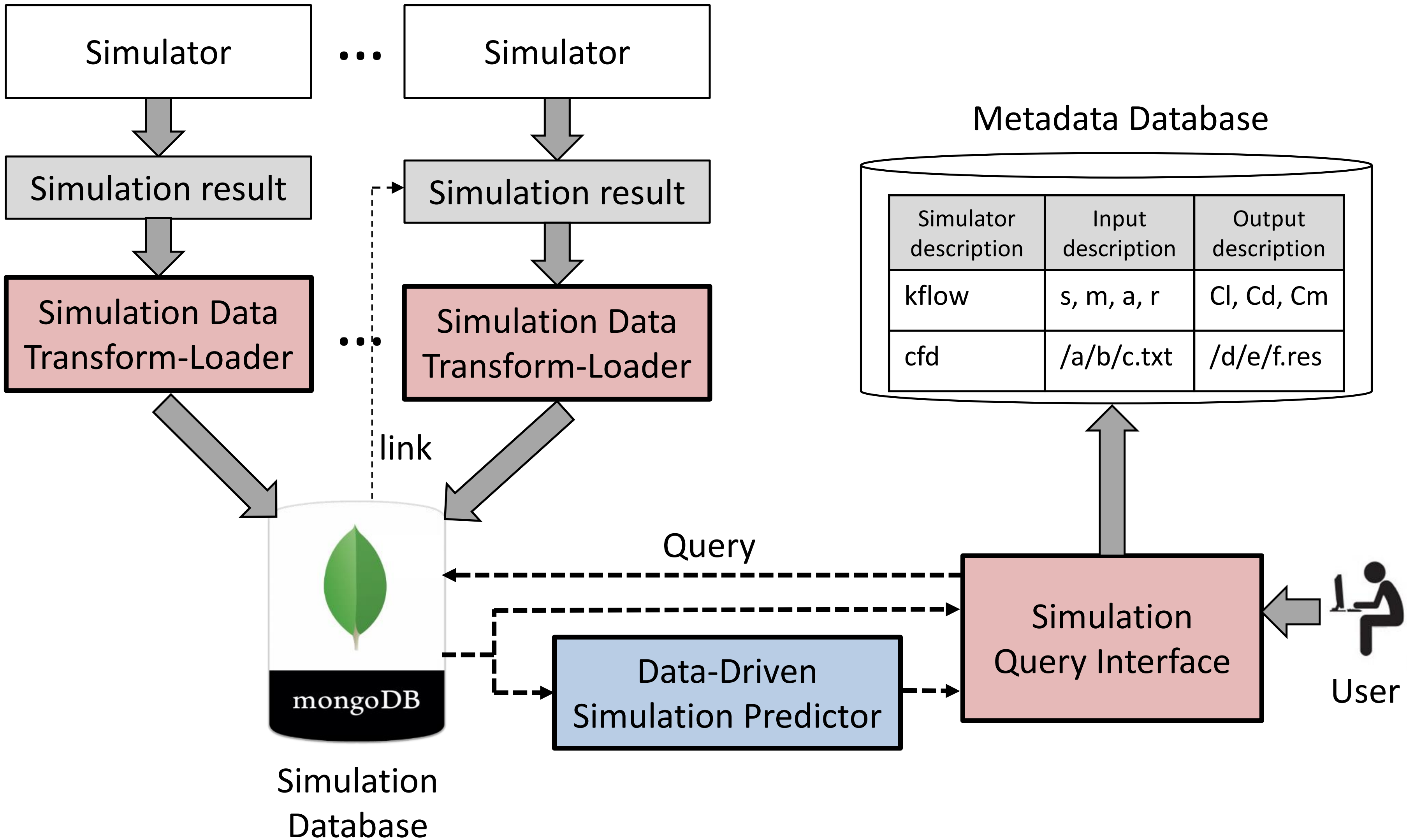
Existing Simulation Service Systems

- iBIOMES
  - A storage and querying system for large biomolecular simulation and computational chemistry datasets
- Scibox
  - A cloud-based simulation data sharing and storage system providing a Dropbox-like interface

✓ **Limitations of existing simulation service systems**

- They do not provide the ability to ***reuse*** the existing simulation results automatically without user involvement
- They do not provide the ability to ***predict*** the result of a simulation based on the previous simulation results

Developed System Architecture

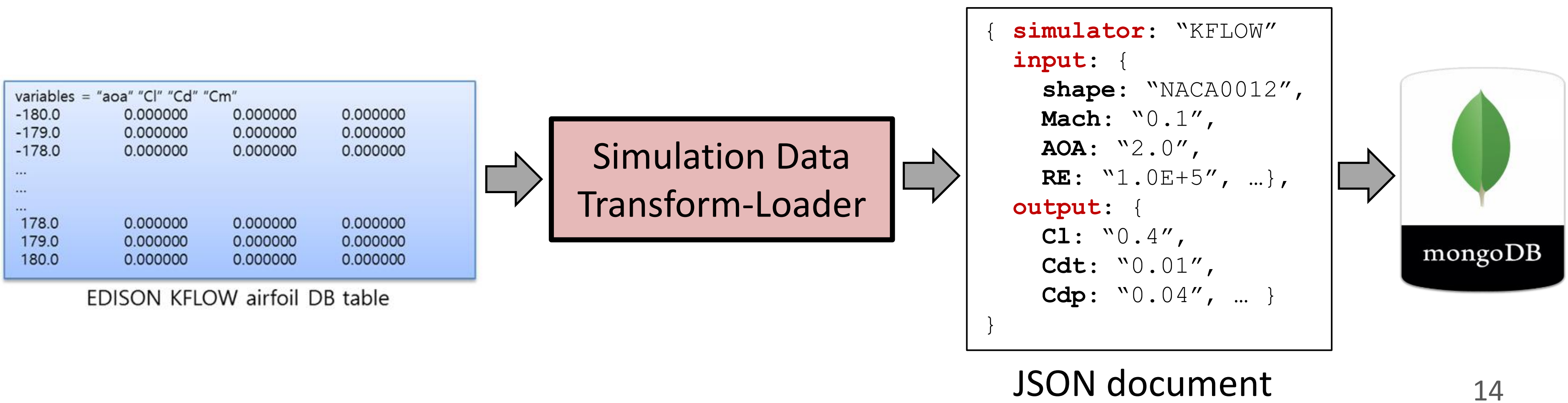


Main Components

1. Simulation data transform-loader
   – Transforms simulation results into JSON documents and loads it into the simulation database
2. Simulation database
   – Stores simulation results
3. Simulation query interface
   – Receives a user request and returns the simulation result to the user
   – If the requested result is in the database, returns the result to the user
   – Otherwise, allows the user to predict the simulation result
4. Data-driven simulation predictor
   – Predicts the result of a simulation based on the previous results
5. Metadata database
   – Stores the information about simulation programs

1. Simulation Data Transform-Loader

- Allows the user to ***load*** the result of a completed simulation into the simulation database
  - Makes a JSON document from the output files and loads it into DB
- Needed for each specific simulation program
  - Because different simulation programs have different result structures
- Example: simulation data transform-loader for EDISON KFLOW



2. Simulation Database

- We adopt ***MongoDB*** as the simulation database

- MongoDB
  - A open-source document-oriented NoSQL database
  - Supports storing JSON documents with dynamic schemas

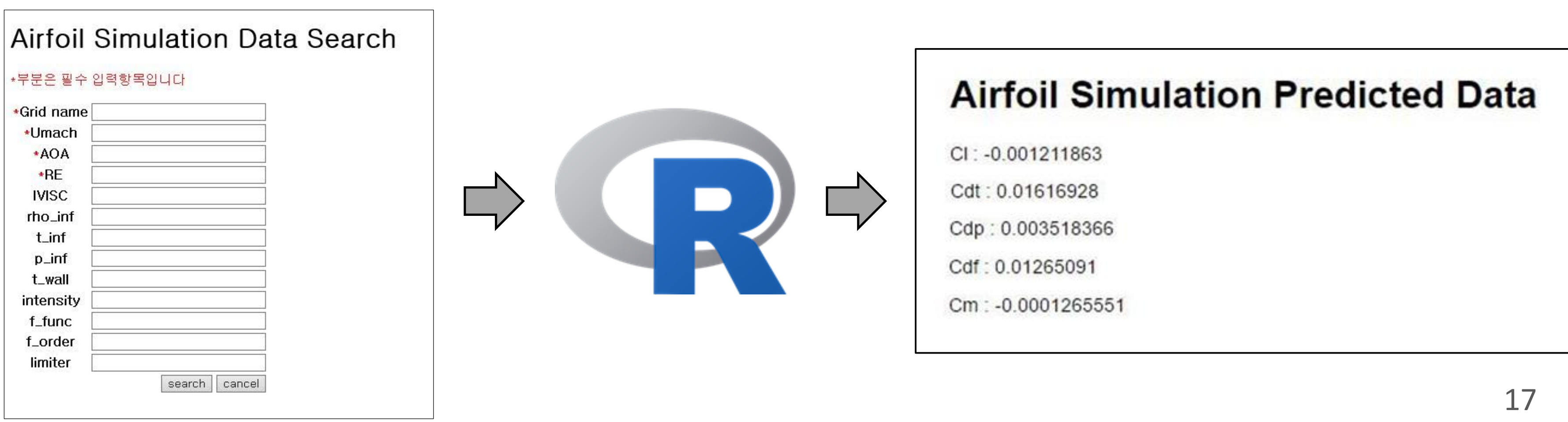
- Two desirable features of MongoDB for our system
  - It supports ***dynamic schemas***
    - We can insert data of any structure without a predefined schema
    - Thus, it is easy to store simulation results with various structures
  - It supports ***storing large volumes of data*** on a large cluster
    - Because the volume of simulation data is rapidly growing, this scalability is quite important for our simulation service system

3. Simulation Query Interface

- Receives a user request and returns the result to the user
  - If the result of the requested simulation exists in the database
    - Returns the found result back to the user
  - If not, provides the user with two options:
    ① Execute the requested simulation from scratch
    ② Predict the result of the requested simulation ***without*** executing it

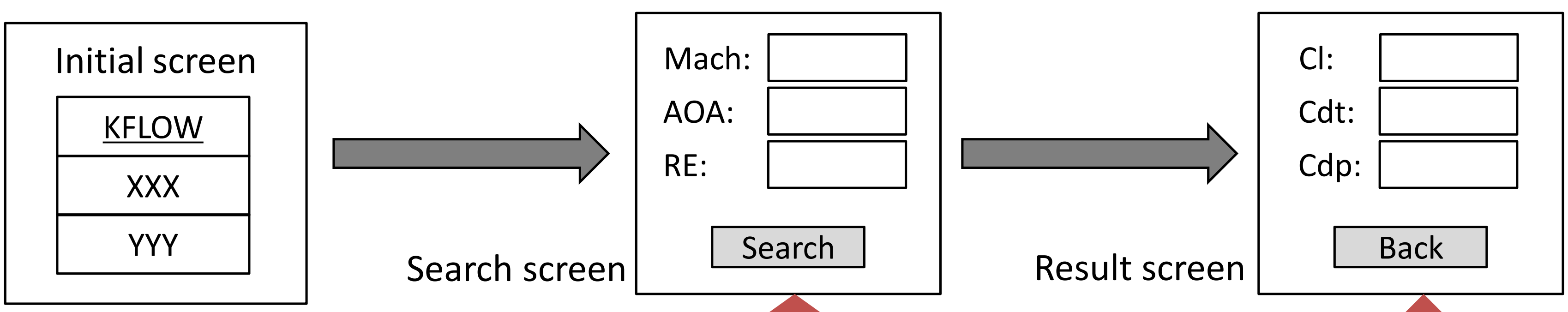
4. Data-driven Simulation Predictor

- The end goal is to reduce the burden on the system
  - We need not to execute the requested simulation actually
- To predict the result of a simulation, we employ a number of ***statistical machine learning*** techniques
  - Linear regression, support vector machine, CART, MARS, local regression, k-nearest neighbor regression, neural networks, etc.
- We used ***R*** library to implement the prediction methods



5. Metadata Database

- One important requirement for our simulation service system
  - To support ***various*** simulation programs
- Thus, we store the metadata for each simulation program
  - **Simulation program**: name, version, etc.
  - **Input parameters**: name, datatype, required or optional, etc.
  - **Output parameters**: name, datatype, display option, etc.



Metadata database

Simulator	Input parameters	Output parameters
KFLOW	**Mach, AOA, RE**	**Cl, Cdt, Cdp**
XXX	…	…
YYY	…	…

Prediction Performance Evaluation

- Dataset
  - EDISON KFLOW simulation dataset (provided by KISTI)
    - **Input parameters**: thickness, Mach number, angle of attack, Reynolds number
    - **Output parameters**: Cl, Cdt, Cdp, Cdf, Cm
    - **The total number of records**: 7680

- Prediction model training
  - The number of records in the training data: 6200 (80% of all data)
  - The number of records in the test data: 1479 (20% of all data)
  - We use 10-fold cross validation (i.e., 9:1 partition for the training data)

- Prediction models
  - Multiple linear regression, GAM, SVM, CART, random forests, GBM, MARS, local regression, k-nearest neighbor regression, neural networks

Evaluation Results

Prediction Model	Cl	Cdt	Cdp	Cdf	Cm
Multiple Linear Regression	12.6%	46.3%	153%	10.9%	59.3%
Generalized Additive Model (GAM)	10.6%	41.1%	130.0%	8.7%	65.1%
Support Vector Machine (SVM) regression	3.7%	6.9%	19.6%	2.5%	21.4%
Classfication and regression trees (CART)	5.8%	7.2%	15.2%	3.5%	17.4%
Random Forests	**0.9%**	**1.6%**	**2.6%**	**1.2%**	**7.5%**
Generalized Boosted Model (GBM)	23.7%	3.4%	7.8%	2.2%	18.3%
Multivariate Adaptive Regression Spline (MARS)	3.9%	9.3%	20.7%	3.7%	30.9%
Local Regression	**1.3%**	**1.7%**	**2.7%**	**1.1%**	**7.4%**
k-Nearest Neighbor (k-NN) Regression	**3.0%**	**3.7%**	**6.5%**	**2.0%**	**10.3%**
Multilayer Neural Networks	**2.1%**	**2.8%**	**13.8%**	**3.0%**	**10.6%**

– Note that we use ***the average relative error*** (= |true – estimate|/true·100%) as the performance measure, instead of root-mean-square error (RMSE)

Result Analysis

- Nearest-neighbor based regressions show relatively ***good*** performance
  - Local regression, k-nearest neighbor regression
- Regressions that produce a single prediction function covering the whole dataset show relatively ***poor*** performance
  - Multiple linear regression, generalized additive model (GAM)
- The performance of decision trees based regressions improves as the number of trees included in the model increases
  - CART, GBM < Random Forests
- It appears that 1 or 2 are enough for the number of hidden layers for multilayer neural networks

Conclusions

- In this paper, we designed and implemented a ***data-driven*** simulation service system
- Unlike the existing systems, our system ***utilizes the previous simulation results*** to improve the execution of later simulations
  - ***Reuse*** of the previous simulation results
  - ***Prediction*** based on the previous simulation results
- Advantages of our system
  - Redundant or unnecessary computation is avoided, resulting in a reduced response time and saved computing resources
  - Consequently, a greater number of users can be served with less amount of computing resources
- We hopes that many scientists and engineers will utilize their simulation results more effectively using our system

Thank you!

Any Question?