

# Development and Performance Analysis of a Simulation-Optimization Framework on TeraGrid Linux Clusters

**Baha Mirghani, Michael Tryby, Ranji Ranjithan and Kumar Mahinthakumar**  
*Department of Civil, Construction & Environmental Engineering  
North Carolina State University*

**Derek Baessler and Nicholas Karonis**  
*Department of Computer Science  
Northern Illinois University*

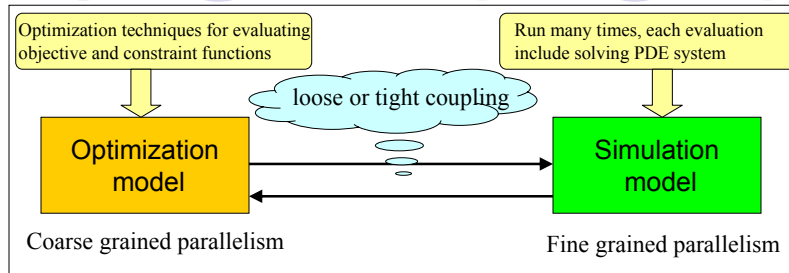
The 6<sup>th</sup> LCI International Conference, Chapel Hill, NC  
April 27<sup>th</sup>, 2005

## Outline

- Introduction
- Optimization Model
- Simulation Model
- Timing Study Experiments and Results
- Conclusions and Recommendations
- Ongoing Work

## Introduction

### Simulation-Optimization



- Simulation optimization is an expensive procedure that needs adequate computing resources.
- Applications: engineering design optimization, model calibration, solution of inverse problems, etc...

3

## Main Objectives

- Generic large scale optimization framework that accommodates different simulator models "different application".
- User friendly optimization framework, with various optimization techniques.
- Optimization framework with acceptable performance and scaling efficiency.

4

## Approach

- Design a large scale optimization framework.
- The framework coupled with a simulation model to solve Groundwater inverse problem "as an application".
- The framework performance benchmarked against an optimization framework developed previously.
- A performance study conducted on TeraGrid Linux Clusters.
- The goal is to conduct several performance optimizations to improve the performance of the current framework.

5

## Current Optimization Framework- LASSO

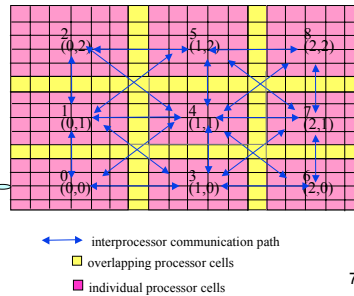
- LASSO (LAarge Scale Simulation Optimization framework) is designed in a modular fashion to simplify coupling with any simulation model "LOOSE COUPLING".
- LASSO framework is currently under development, and has been implemented using Java programming language.
- LASSO is portable and has been tested on various supercomputing resources including Linux Clusters.
- LASSO framework has been developed as a **generic optimization tool** useful for **many environmental application areas**: Groundwater, Surface Water, Air quality etc.
- LASSO is designed to manage multilevel parallelism for efficient parallel speedup.

6

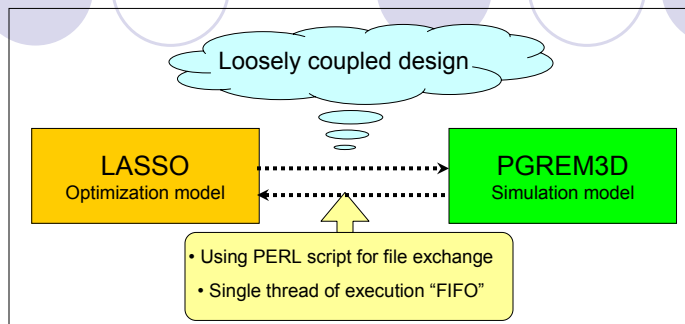
## Test Simulation Model- PGREM3D

- PGREM3D (Parallel Groundwater transport and REMediation) is a parallel Finite Element code used for numerical simulation of 3D groundwater flow and transport problems.
- The codes are written in Fortran. It has been tested extensively for scalability and performance on different parallel architectures.
- The simulator is parallelized using 2D domain decomposition. MPI library is used to exchange information between these domains.

Fine grained parallelism



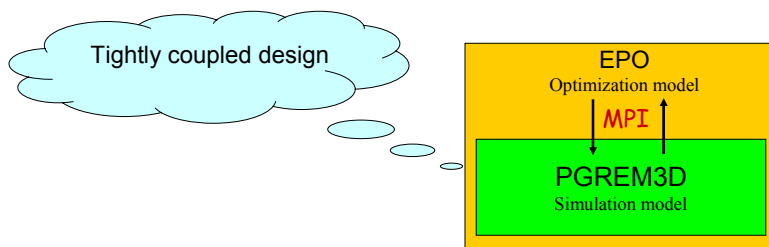
## LASSO - Simulator Coupling



- Supports heuristics searches (genetic algorithms and evolutionary strategies) and direct local search techniques (Nelder-Mead Simplex)
- Development of gradient based procedures, hybrid search techniques, and distributed genetic algorithms are planned.

## Previous Optimization Framework- EPO used as a reference

- EPO (Efficient Parallel Optimization framework) is implemented in Fortran & MPI for solving problem with multilevel parallelism (Sayeed 2004).
- Our intention is not an outright performance comparison between Java and Fortran MPI, since we anticipated Java to be slower in the first place.



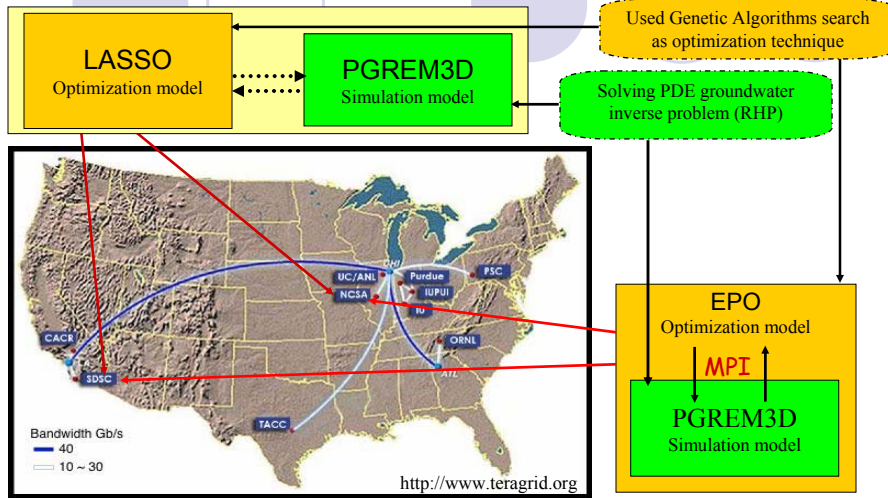
9

## Linux Clusters Platforms

- NCSA TeraGrid
  - Processor: Itanium2/IA-64, 1.5 GHZ.
  - Performance: 6 Gflops in Peak.
  - Network: Myrinet 2000, Gigabit Ethernet, Fiber Channel.
- SDSC TeraGrid
  - Processor: Itanium2/IA-64, 1.5 GHZ.
  - Performance: 4 Gflops in Peak.
  - Network: Myrinet, Gigabit Ethernet, Fiber Channel.

10

## Timing Study Experiments

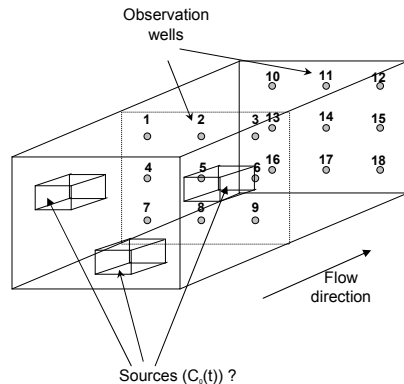


- A performance study conducted on TeraGrid machines by doubling the number of processors (1, 2, 4, ..., 128).

## Test Application:

### Groundwater Source Release History Reconstruction Problem (RHP)

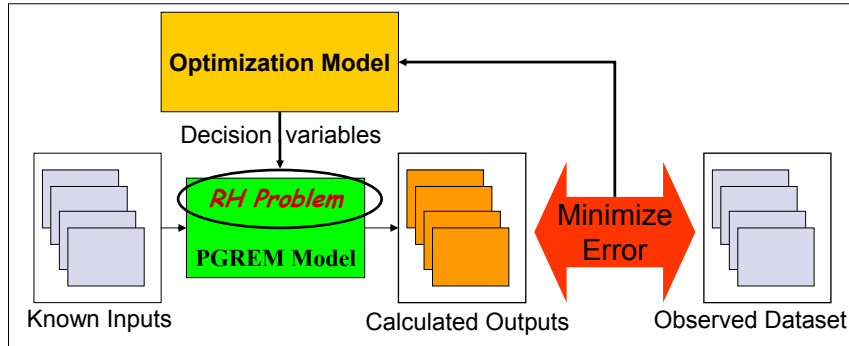
- Location of contaminant sources are known.
- Temporal release histories of contaminant source(s) are unknown.
- FEM problem size is  $51 \times 31 \times 11$  (17,391 nodes) and 1000 time steps.



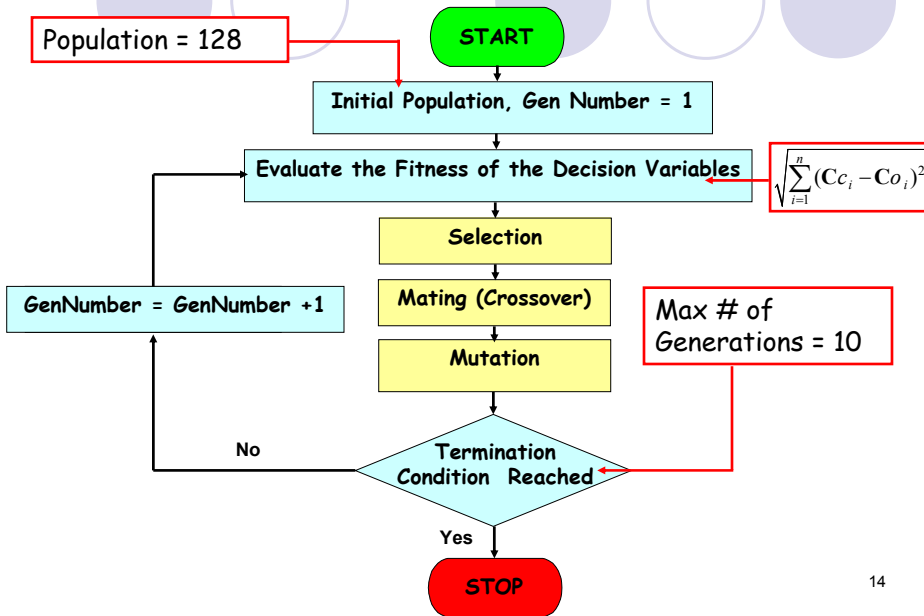
# Simulation-Optimization Inverse Modeling for (RHP)

- The inverse problem is posed as an optimization model.  
 where the objective function is to minimize (RSE):

$$\sqrt{\sum_{i=1}^n (Cc_i - Co_i)^2}$$



# Genetic Algorithms Process



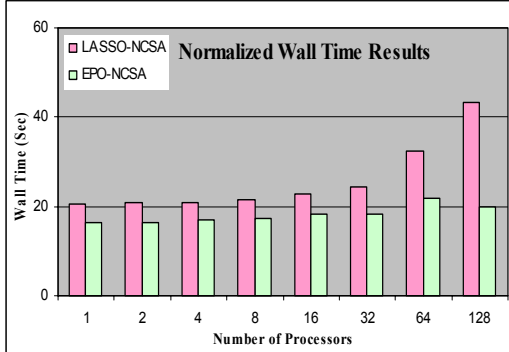
## Timing Study Run Settings

- The Simulator (PGREM3D) executable was compiled and run on a single processor.
- This enabled us to isolate the coarse grained parallelism within the solution procedure for performance benchmarking.

## Performance Results

## Timing Study Results on TeraGrid NCSA

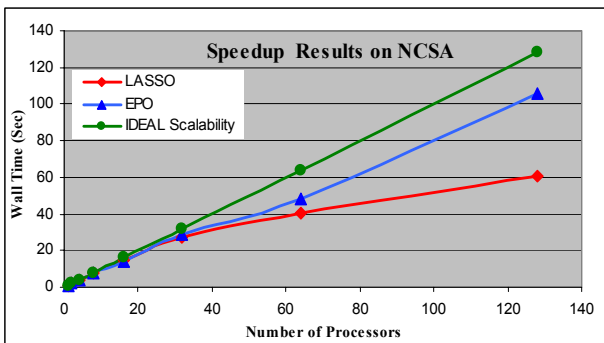
Number of Processors	Time (sec)		Time for one Eval. per one Proc. (sec)	
	LASSO	EPO	LASSO	EPO
1	19186	20954	20.45	16.37
2	9833	10559	20.97	16.50
4	4911	5459	20.94	17.06
8	2537	2792	21.64	17.45
16	1329	1473	22.67	18.41
32	711	732	24.26	18.30
64	474	434	32.34	21.70
128	318	198	43.39	19.80



LASSO has 938 evaluations while EPO has 1280 evaluations.

## Speedup Results on TeraGrid NCSA

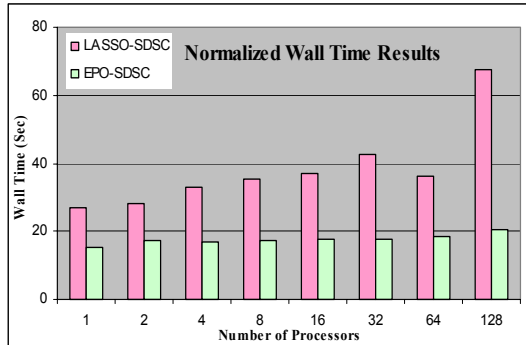
Number of Processors	Scalability	
	LASSO	EPO
1	1.00	1.00
2	1.95	1.98
4	3.91	3.84
8	7.56	7.51
16	14.44	14.23
32	26.98	28.63
64	40.48	48.28
128	60.33	105.83



File I/O contention contribute negatively to LASSO speedup when number of processor was greater than 64

## Timing Study Results on TeraGrid SDSC

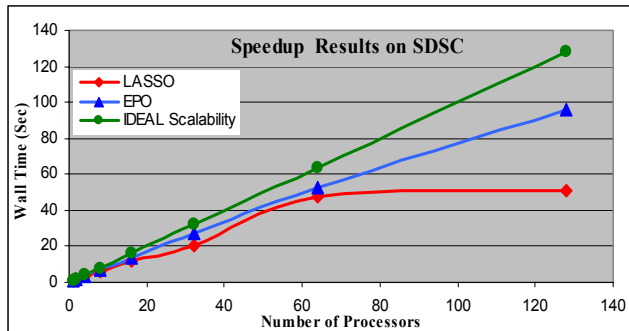
Number of Processors	Time (sec)		Time for one Evaluation per one Proc. (sec)	
	LASSO	EPO	LASSO	EPO
1	25220	19503	26.89	15.24
2	13213	11015	28.17	17.21
4	7686	5407	32.78	16.90
8	4155	2756	35.44	17.23
16	2172	1409	37.05	17.61
32	1249	714	42.61	17.85
64	529	369	36.09	18.45
128	494	204	67.41	20.40



LASSO has 938 evaluations while EPO has 1280 evaluations.

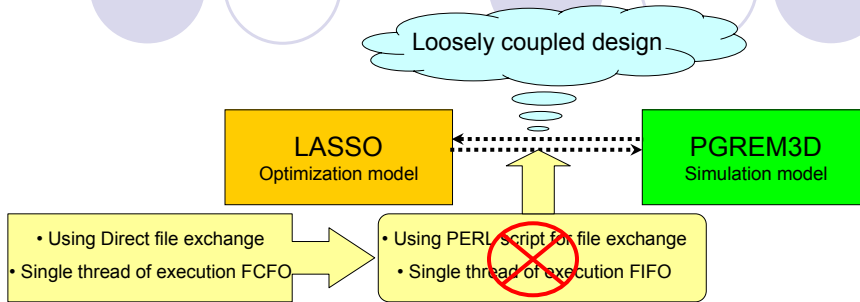
## Speedup Results on TeraGrid SDSC

Number of Processors	Scalability	
	LASSO	EPO
1	1.00	1.00
2	1.91	1.77
4	3.28	3.61
8	6.07	7.08
16	11.61	13.84
32	20.19	27.32
64	47.67	52.85
128	51.05	95.60



File I/O contention contribute negatively to LASSO speedup when number of processor was greater than 64

## Performance Improvements



- Direct file exchange between the LASSO and the PGREM3D executable instead of the PERL script that used to handle the file exchange.
- Handle tasks as they are completed instead of handled on a first in first out basis.

## LASSO Performance Improvement Results on TeraGrid NCSA

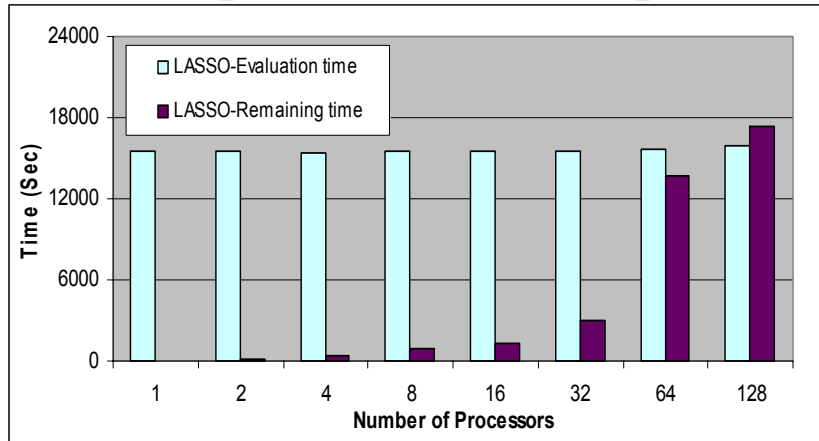
Number of Processors	Time (sec)		Time for one Evalua. per one Processor (sec)		Speedup	
	Previous	Modified	Previous	Modified	Previous	Modified
1	19186	15565	20.45	15.57	1.00	1.00
2	9833	7797	20.97	15.59	1.95	2.00
4	4911	3946	20.94	15.78	3.91	3.94
8	2537	2047	21.64	16.38	7.56	7.60
16	1329	1052	22.67	16.83	14.44	14.80
32	711	577	24.26	18.46	26.98	26.98
64	474	396	32.34	23.62	40.48	39.30
128	318	259	43.39	33.15	60.33	60.10

20%

20%

Same

## LASSO Evaluations time VS. Remaining time on TeraGrid NCSA



Remaining time = (Total Time - Evaluations Time)

23

## Observations and Conclusion

- Communications overhead contributes significantly to evaluation times especially when the numbers of processors was greater than or equal to 32.
- File exchange between the executable server and their simulation executables make model evaluations more costly in the LASSO framework.
- The ratio of Java/Fortran wall times in this study ranged from 1.2 to 3.2, we believe this is an acceptable level of slow down given the advantages of the LASSO framework design.

24

## Recommended Performance Optimizations

- Eliminate file-based communication
  - Use MPICHG2 calls instead of the file exchange.
- Use of higher speed network
  - Directing communications between the LASSO and its evaluation servers over high speed network (Myrinet) instead of Java TCP-IP socket communications.

25

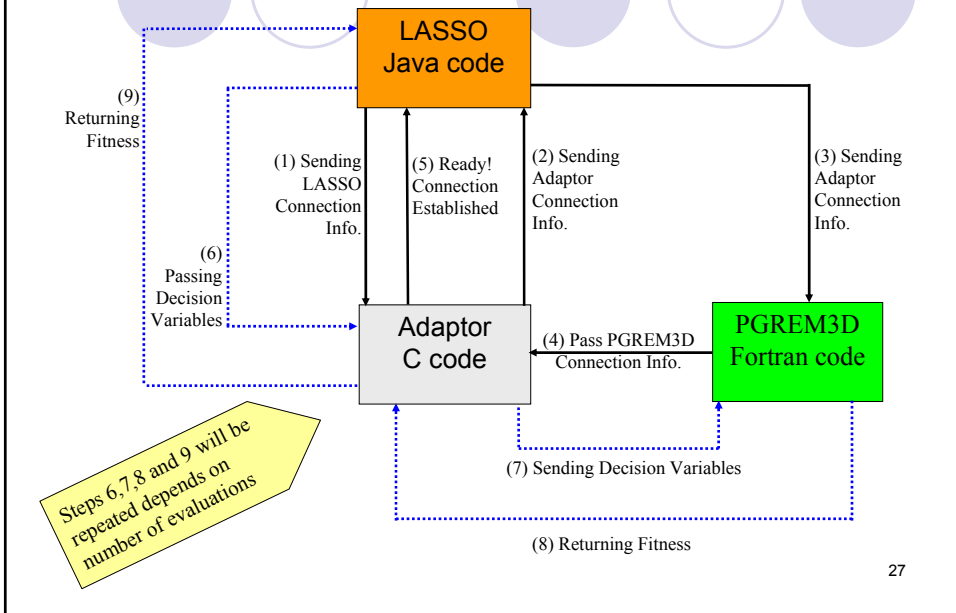
## Ongoing Work

- Eliminating file exchange and creating a generic interface for decision variable and evaluation result exchange using MPICHG2.

Construct an adaptor process written in *C* language that translates Java objects into native variables compatible for MPICHG2, and sends them via a socket connection with the executable established via an MPICHG2 call.

26

## LASSO Framework using MPICH2



27

## Acknowledgement

- This work was supported by National Science Foundation (NSF) under Grant No. **BES- 0238623**. The authors are grateful for the TeraGrid supercomputer resources provided by NCSA and SDSC.
- Grateful to the co-authors for their valuable contributions.

28